Author version, published in: ***Interacting with Computers*, volume 12, issue 3, pp. 245-279, 2000.**

A model based approach to semi-automated user interface generation for process control interactive applications

Faouzi Moussa [a], Christophe Kolski [b], Myriam Riahi [a]

[a]*Ecole Nationale des Sciences de l'Informatique de Tunis, rue des Entrepreneurs Charguia II, 2035 Tunis-Carthage, Tunisia*
[b]*Laboratoire d'Automatique et de Mécanique Industrielles et Humaines - UMR CNRS 8530, Le Mont Houy, BP 311, 59304 Valenciennes cedex, France*

Abstract

The purpose of this paper is the description of a model-based approach (called ERGO-CONCEPTOR) to semi-automated user interface generation. Its application field concerns process control interactive applications. From a database describing the industrial process (for instance a chemical process) to be supervised, the system is able to generate automatically UI specifications using guidelines. The specifications contain eventually, design alternatives. They can be selected by the designer who can then interactively generate the UI. ERGO-CONCEPTOR is composed of three main modules. The first allows an interactive process description according to three sub-models (a physical model, a structural model and a functional model). The second module uses a knowledge-based approach for the automated generation of the UI specification. Finally, the third module is used by the designer to interactively generate the specified UI. © 1999 Elsevier Science B.V.

Keywords: Automatic generation, guidelines, human-computer interaction, human factors, interactive application, interface design, model-based approach, process control, software ergonomics, specification, user interface (UI).

1. Introduction

During recent years, the development of new technologies in complex industrial systems (such as chemical industries, nuclear processes, and transport systems) has modified human tasks into high level tasks: system management, supervising and decision making in critical situations. The variety of strategies, which can intervene in the process in case of dysfunction, can be enormous. It is why these complex tasks can not be managed by automated systems. Thus, the operators' responsibilities have increased. Human errors can have dangerous consequences for the production system, environmental safety and human lives.

The current tendency in control rooms consists of presenting data on graphical screens. But many methodological problems have appeared. Indeed, because of the complexity of many industrial processes (a complex process can be composed of hundreds, or thousands, of dynamic variables), interface design and evaluation is considered to be very long and difficult, since process control interactive applications exist. Despite much research in this field (see for instance [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]), there exists no systematic and automatic approach for interface development.

Since the eighties, a research way studied by some researchers concerns knowledge-based approaches for automatic evaluation and design of UI used in process control [12, 13, 14, 15, 16, 17]. To our knowledge, few knowledge-based prototypes have been tested in the field of process control. To be really efficient, such research approaches must necessarily include contributions from various disciplines: computer science, software engineering, artificial intelligence, task analysis, human factors, and so on. The ERGO-CONCEPTOR system described in this paper is oriented towards process control interactive applications. For that reason, it is also necessary to consider other knowledge and principles pertaining to automation; some of which will be progressively described in the paper (particularly concerning the domain description, i.e. the industrial process description).

According to the principles described by Myers [18] or Szekely [19], ERGO-CONCEPTOR can be considered as a model-based approach for user interface development. ERGO-CONCEPTOR is at present a prototype. It is written in LISP. This system allows the interface design to be based upon an automatic generation of UI specifications [17].

In this paper, our research works are stated and the main principles on which ERGO-CONCEPTOR is based are explained: particularly, we describe the concerned application field (process control) in addition to the basic principles of model-based interface development. Then the software architecture is described. Finally, a technical validation of ERGO-CONCEPTOR is presented.

2. Positioning of the research and principles used by ERGO-CONCEPTOR

2.1. ERGO-CONCEPTOR is dedicated to process control interactive applications

First, it is necessary to provide the global presentation of the process control field. Then, a global design and evaluation methodology usable in the process control field is explained. ERGO-CONCEPTOR will be located in the global methodology.

2.1.1. Global presentation of the process control field

In this paper, we are interested in graphic UI used in the control rooms of industrial complex systems (such as chemical processes, nuclear processes, power networks, and automated transportation services). In these control rooms, the human tasks are often complex; the decision criteria used by human operators are various. These criteria relate to production, safety, economy, quality, ecology, and so on, and are applied in dynamic and diversified situations [2, 4, 7, 10, 20]. The UI plays an essential role in the efficiency and the reliability of the human-machine system. In control rooms, the human operators are generally far from the industrial installations. They have to solve problems involving hundreds or thousands variables (such as temperatures, flows, pressures, etc). The human tasks require a high level of knowledge and know-how. These have been grouped by Rouse [21] in four main classes:

- *The transition tasks:* these correspond to particular pre-defined states of the process (start, change of functioning point, stop, etc). The UI must simplify (1) the performing of pre-defined procedures, (2) the judgement concerning their repercussions on the process and (3) the supervision of the process evolution.
- *The supervision and optimization tasks:* the UI must simplify the supervision of the current process state, in order (1) to detect and anticipate the appearance of abnormal events, and (2) optimize the production using fine adjustments.
- *The default detection and diagnosis tasks:* due to alarm appearance and/or the possibility of observing abnormal evolution of certain process variables, the human operators must be able to detect defaults (failures) and to diagnose the problem. It is very useful if the UI shows the cause/effect relationships between the process variables.
- *Compensation and correction tasks*: in order to re-establish the normal functioning of the process, the UI must help the human operators to decide which actions to perform. The UI must also provide the ability to study the effects of these actions on the process.

To perform complex tasks, the human operators use UI to graphically present a huge amount of information. The UI can be associated with assistance modules. Such modules can be "intelligent" (ie, based on Artificial Intelligence techniques) or typically algorithmic: they are intended to assist with alarm filtering or management, diagnosis, default prediction, simulation of actions, and so on [2, 3, 11, 22, 23].

UI must synthetize the current state of the process and support the human operators during their activities. The human operators do not have a direct view of (1) the industrial process and (2) the results of the actions. At a distance, they must build and refresh in real time (through the UI) a mental representation of the process and of the evolution of the main variables of which it is composed [2, 24].

The UI is centralized on one or several visualization screens; it is comprised of sets of displays, sometimes hundreds or thousands of displays, according to the process complexity. Each graphic display can show several a multitude of different information. A display is composed of two distinct parts: (1) a static part (also called "background") which contains information that cannot be modified, and (ii) a dynamic part displaying animated information to allow the study of quantitative and qualitative variations of the process variables. Many industrial graphic environments for developing process control interactive applications are available on the international market. These allow (more or less, easily) the building of the static part with graphic functions (rectangles, vectors, texts, and so on). With regard to the dynamic part, they propose different types of more or less advanced graphic functions: animated symbols, messages, bargraphs, dials, trends, numeric counters, and so on. Kolski [11] explains many traditional and advanced representation modes.

Due to the huge number of variables to consider and the complexity of the tasks to perform, the use of UI in control rooms can lead to human errors. These errors occasionally result in some catastrophic consequences for humans, the installations, the production and/or the environment (Three Mile Island, Chernobyl, aircraft crashes, and so on). It should be noted that literature contains many classifications and studies of human error (see for instance [25, 26, 27]). Human errors can result from insufficient analysis of the user requirements for the performing of different tasks. Other cases that contribute to human error include insufficient consideration of the cognitive resources and limits of users, deficiencies in information presentation and structure on screens, mal-adjusted definition of UI architecture, insufficient evaluations in simulated or real situations, and so on.

2.1.2. Global design and evaluation methodology usable in the process control field

The aim of this section is to explain, globally, a design and evaluation methodology to be used in the process control field (with the final goal of better locating ERGO-CONCEPTOR). This presentation resumes different works (for more details see [3, 22, 24, 29]). This methodology could also be integrated in a more general methodology, of complete control room design [31, 32, 33]. The methodology consists of several major stages, as highlighted in figure 1. This

description, however, details in particular the human-machine system analysis and modelling stage. In fact, this stage precedes the use of ERGO-CONCEPTOR. [1]

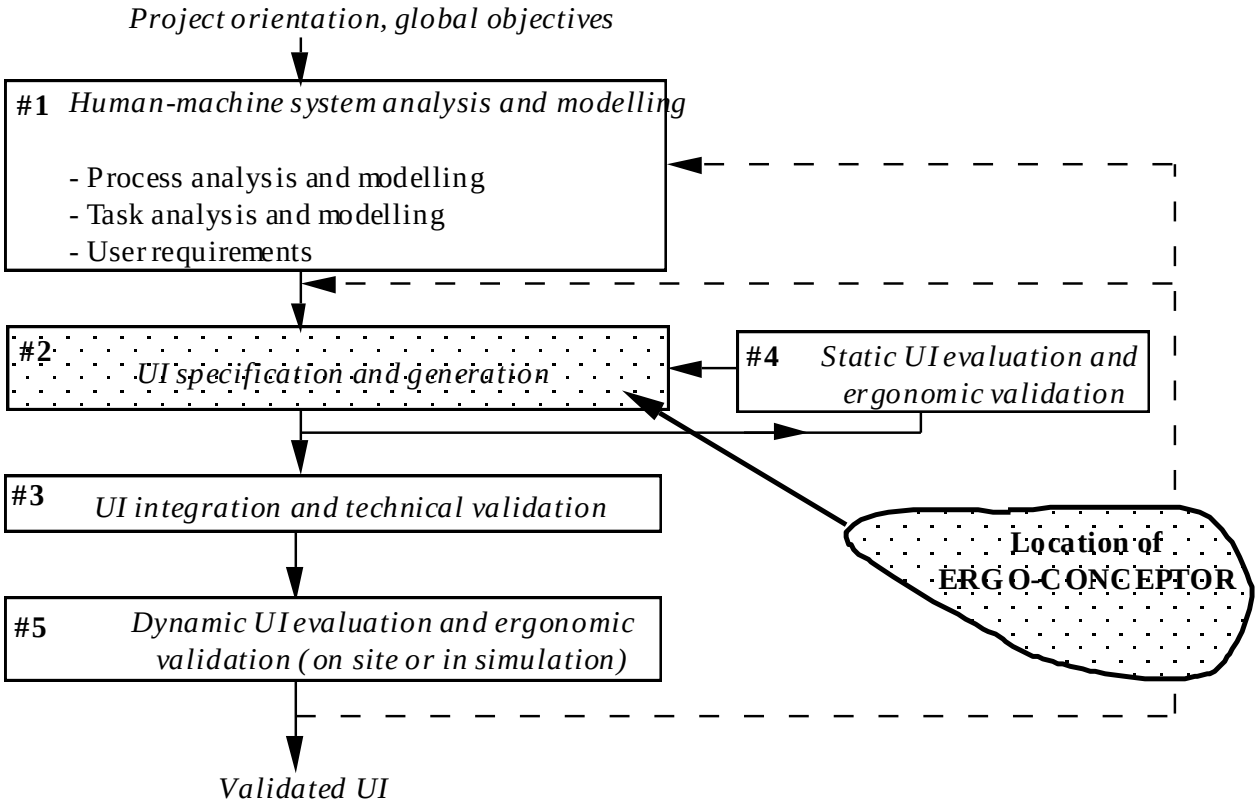


Fig. 1. Global design and evaluation methodology

Some preliminary work is required within **the human-machine system analysis and modelling** (first stage in figure 1). This work consists of constructing a database containing data about the process (main sub-systems, variables, relationships between sub-systems and/or variables, types of relationships, and so on; for instance a sub-system called "machine5" can been composed with different variables, such as temperature-3 and pressure-P89). The technical documentation of the process is an indispensable part of the design of every system. In the context of our work, the UI design assumes that the physical process already exists or at least that an analysis to support its design has already been completed. This analysis is usually undertaken by means one of the numerous existing techniques that allow the description of the structure of the process, its normal functioning and its abnormal functioning. Another assumption is that this can be done at least at the level of complexity and precision needed for this analysis. One could use methods that already exist, and which can be assumed to be familiar to engineers [11, 34]. Some of these methods are oriented towards normal situations of the process. For instance, there are currently well known, structured methods such as SADT (Structured Analysis and Design Technique [35, 36]) or SA (Structured Analysis [37]). These are based upon approximatively the same principles, with a description on different abstraction levels. Functional methods are also available, for instance the graph fluence method [38], where a sub-system is described by variables in relation. Other methods are oriented towards abnormal situations, such as the formal Petri Nets method, or the textual FMEA (Failure Mode and Effects Analysis [39]) method, in which the possible defects of a system are listed in tables, and associated with potential consequences, gravity rate, and so on.

In addition, to complete this technical database, it has become recognized in recent years that a human task analysis is necessary for the success of the project. An appropriate task analysis provides information requirements about the human operators. These requirements must be carefully considered during the design phase of graphic interfaces [2, 40]. It is well known that, in process control, due to a large number of dynamic variables, the abnormal situations can be extremely various, and can lead to incidental or accidental situations. Such situations can be catastrophic for human beings involved, and the environment, particularly in chemical processes and transportation systems. In these conditions, it is obviously very important to have full knowledge of the human requirements in case of abnormal situations [26, 33, 41].

A further source of information that must be included concerns the control and command objectives that represent an extra dimension needed to define the system. This will ensure that operators will be able to satisfy the particular goals identified by the design engineers. Thus, in this stage at least the following elements will be specified: sub-systems of

[1] At our sense, the global methodology could be used in many other application fields where UI are required. According with a considered application field, specific methods, tools, modes of presentation or criteria must then to be taken into account by the designers.

the process to control, the different types of people having to intervene in particular contexts, changes which need to occur in graphical representations of system state dependant upon the particular contexts and system states. The overall results of this stage of analysis provide all the data needed in the next stage.

The UI specification and generation stage (second stage in figure 1) can be used to obtain a framework of screens deduced from the structural and functional decompositions of the operator's informational needs. For instance, one of the screens must provide the human operator with a concrete representation of a cooking system; another must display the values of ten temperatures and five pressures. The modes of graphic information presentation for every screen must also be specified. Thus, animation functions are chosen, and associated with a static graphic environment, color-coding, etc. For instance, the function, which displays a curve, is placed on the center of the screen, associated with a title, graduations and axis. Furthermore, the color of the curve display could be green if the variable is in normal operation mode or red if the variable is in abnormal operation. In this stage the graphic creation of the displays is made by means of an editor of animated graphic control displays. In most cases, the UI generation is manual in this stage. *We propose to use ERGO-CONCEPTOR in this stage for a semi-automated generation of a first version of the UI.*

The next stage consists of the UI integration in the control room, the connection with different calculators and sensors located on the process, and the technical validation of that integration (third stage in figure 1).

Ergonomic evaluations aim to ensure that the displays thus created answer the informational requirements of human operators. These evaluations can be divided into two other stages : the "static" and "dynamic" ones.

The objectives of the **"static" evaluation stage** (fourth stage in figure 1) are (1) to verify with the human operators that the displays provide adequate information, and (2) to verify that the UI respects the ergonomic rules of information presentation on a screen. For this second objective, it is necessary to apply a set of ergonomic criteria related to the human information acquisition capacities. These criteria are concerned with:

- the accuracy of the representation modes chosen for each piece of information;
- the shape of the different symbols used to represent the components of the process; for instance: specific colors for specific types of information, contrasts between these colors;
- the localization of the different items composing each display, and
- the arrangement of the windows on the screen.

The ergonomic criteria applied are generally expressed in terms of rules in design guidelines (see 3.2). *For this stage, we propose to integrate a set of rules in the ERGO-CONCEPTOR knowledge base: this knowledge base is in fact used a priori directly during the stage #2.*

The "dynamic" evaluation stage (fifth stage in figure 1) is carried out on the industrial site and/or in simulation. Its principal aim is to verify that the structure and content of each display responds to the operator's informational needs during the execution of tasks, and in accordance with the different normal and abnormal contexts of the process.

For these two evaluation stages, there are many evaluation methods that can be used, such as observations, protocol analysis, questionnaires, and so on [11, 42, 43, 44]. These two ergonomic stages lead either to a validated interface or returning to the previous stages to improve the final system (see figure 1). An aim of this global methodology is the taking into account of human factors in the design loop. These factors are particularly treated in the "static" and "dynamic" ergonomic stages. But it is necessary to add that the actual tendencies of the designers, which lead to recommend the screens for unique information support, must be considered with care. Indeed, in the information system used by operators, a place must be reserved for verbal communications and their functional utility [20].

2.2. ERGO-CONCEPTOR is a model-based approach for user interface development

In this section, the principle of model-based user interface development is first explained. Then the interest in studying a model-based approach for process control interactive applications is discussed.

2.2.1. Model-based user interface development - related work

As explained by Szekely [19], "model-based user interface development tools trace their roots to work on user interface management systems done in the early 1980's."

Model-based UI development consists of an alternative paradigm for constructing interfaces [18]. In fact, developers do not use functions from a toolkit library to program the interfaces: they write a specification in a specialized, high level language, such as state transition diagrams [45], grammars [46], or event-based representations [47]. The specification is then automatically translated into an executable program; the specification can also interpreted to automatically generate the user interface [19].

Several model-based user interface development tools are described in the literature. It is important first to review the type of model upon which they are based, which one of them follows a rule-based approach, what they automatically generate, whether they are targeted at a particular application domain (see table 1), and then to globally compare these existing tools with ERGO-CONCEPTOR. [2]

[2] For more exhaustive and deep studies concerning the existing model-based user interface development tools see [48] or [49].

Table 1.

Global comparison of ERGO-CONCEPTOR with representative model-based user interface development tools

Model-based tool	Type of model	Rule-based approach	Automatic generation	Domain
GENIUS [48]	Data model (extended entity-relationship model)	Production rules (implemented with Nexpert Object, an expert system shell) for selection and layout of interaction objects	User interface	Dialogue user interface
DON [50]	Interface Definition Language	Production rules (with Automated Reasoning Tool, an expert system shell) for selection and layout of interaction objects (associated with evaluation metrics)	Graphical interface language	Dialogue user interface
HUMANOID [51]	Declarative modelling language	Production rules (CommonLisp) for selection and layout of interaction objects ; for help generation	User interface integrating a help component	Dialogue user interface
TADEUS [52]	Dialogue graphs	Production rules for selection and layout of interaction objects	User interface	Dialogue user interface
MECANO [53] MOBI-D [54]	MIMIC interface modelling language	Production rules in MECANO (C++) for selection and layout of interaction objects	User interface	Dialogue user interface
TRIDENT [55]	Data model, function chaining graph	Rules organised in decision tree (with AION/DS, an expert system shell) for selection and layout of interaction objects	User interface	Dialogue user interface
UIDE [56]	Interface Definition Language, frames	Production rules (C++) for selection and layout of interaction objects ; for help generation	User interface integrating a help component	Dialogue user interface
JANUS [57]	Object model	Production rules (C++) for selection and layout of interaction objects	User interface	Dialogue user interface
ERGO-CONCEPTOR	Industrial process Description (physical, functional and structural models)	Production rules (Le_Lisp) for selection and layout of interaction objects	HCI specifications (with design alternatives)	Process control interactive applications

With regard to the type of model, the table 1 shows that different approaches are used from one system to another (for instance an object model in JANUS or an extended entity-relationship model in GENIUS). ERGO-CONCEPTOR uses a specific modelling influenced by the application domain.

According to Szekely [19], a model-based interface development process generally uses a rule-based approach for interface design. In fact, each tool in table 1 is based on such an approach, often for selection and layout of interaction objects.

In most cases (GENIUS, TRIDENT, JANUS...), the user interface is generated (see table 1). In ERGO-CONCEPTOR, only the UI specifications are automatically generated, and these specifications can contain design alternatives. In that context, we consider that the designer must remain **the final decision-maker** during the interactive generation of the user interface. It was the choice we made.

Finally, in contrast to all the other tools, ERGO-CONCEPTOR is specific to a particular application domain. It aims at the generation of process control interactive applications.

2.2.2. The reasons for studying a model-based approach for the process control domain

Due to the complexity of the industrial process and all the normal and abnormal situations which can occur, the process control interactive applications can be composed of hundreds of graphic screens. Therefore it should be obvious that it is necessary to reduce the development effort and to improve the quality of the user interface. Moreover, being

able to rapidly obtain a first version of the user interface can provide the development team with more time to evaluate the user interface in conjunction with users (on real site or in simulation). This is crucial in process control applications where a usability problem can lead to catastrophic situations. The model-based approach is interesting for two main reasons:

1. There are many ergonomic rules in currently available literature. To test a model-based approach for process control interactive applications provides the opportunity to study those that can be used and formalised in a particular domain. This work is, itself, a difficult research theme. For instance, Gilmore et al. [5], inspired by their works with the U.S. Nuclear Regulatory Commission (USNRC), have written a handbook of guidelines for user-computer interface design in process control. These guidelines were organized into four categories, namely, (1) video displays, (2) control and input devices, (3) control/display integration, (4) workplace layout and environmental factors. Each guideline was presented in a structured format. Hundreds of guidelines adapted to this particular field (process control) were listed in the handbook. There are many research questions about these guidelines, such as: which of them can be formalised, why, what are goals (automated or semi-automated design and/or evaluation, integration in a didactic electronic document, integration in a knowledge-based system, etc.), does more or less efficient formalisation possibilities exist (based on artificial intelligence methods or not) ? Of course, many researches, developments and evaluations are required to answer such difficult questions. In this paper, one research solution is proposed.
2. A model-based approach encourages a rigorous preliminary modelling of the industrial process. This preliminary modelling needs a difficult and long work, but it leads to an important database for the overall team.

In the next section, the main stages followed by ERGO-CONCEPTOR are explained.

2.2.3. The three stages followed by ERGO-CONCEPTOR according to a model-based approach

With regard to the phases on figure 1, we propose the following hypothesis. We suppose that we can directly exploit data from human-machine system analysis and modelling. These data must be available; they must include the user (i.e. the human operator) requirements that depend on the different contexts of the process to be controlled. Thus, during normal operation, the operator needs sub-sets of significant variables showing its state of normal functioning. However, during abnormal functioning, some information presented to the controller is derived from decision help tools. This information could include alarms, predicated values of variables, advice about actions to perform, and so on.

To generate a process control interactive application, three stages are carried out using ERGO-CONCEPTOR [17, 58], as defined in figure 2. According to the model-based approach terminology [19], the first stage (#2.1) consists of complimentary aspects of process modelling:

1. **a physical model**, represented by a description of the industrial process with different abstraction levels. For instance, at a very high level, the description of the process is very synthetic, representing for example, its goals concerning production, safety, and so on. At a very low level, sub-systems are described by a set of physical variables, such as concrete temperatures or speeds.
2. **a structural model**, represented by a data flow diagram, where a set of sub-systems is represented; each sub-system is composed of different representative variables.
3. **a functional model** details relationships and influences between variables (for instance, if the temperature #4 increases, the pressure #67 will also increase). This description leads to a first level of data necessary for the UI specification, and must follow a specific method, oriented to user requirements and the control-command goals.

In stage #2.2, the data described in the previous stage are used to **automatically generate the UI specification**. Thus, user requirements correspond literally to UI specifications. The UI specifications propose a set of displays; each display is proposed to be structured into specific zones (title zone, command zone, work zone, and so on). In the UI specifications, informational entities (associated with specific representation modes, for instance curves or bargraphs, and associated also with graphic attributes) make up the zones of each display. **Design alternatives** can be included in some specifications ; for instance, for the representation of a set of six variables representative of the state of a sub-system, different possibilities exist: bargraphs, star view, numeric values, and so on [2, 11, 59, 60]. At this level, an ergonomic knowledge base is used for the choice of the graphical attributes, the representational modes, the process variables and the appropriate display structure. At the end of this stage, a listing of UI specifications is automatically generated, and directly available in stage #2.3.

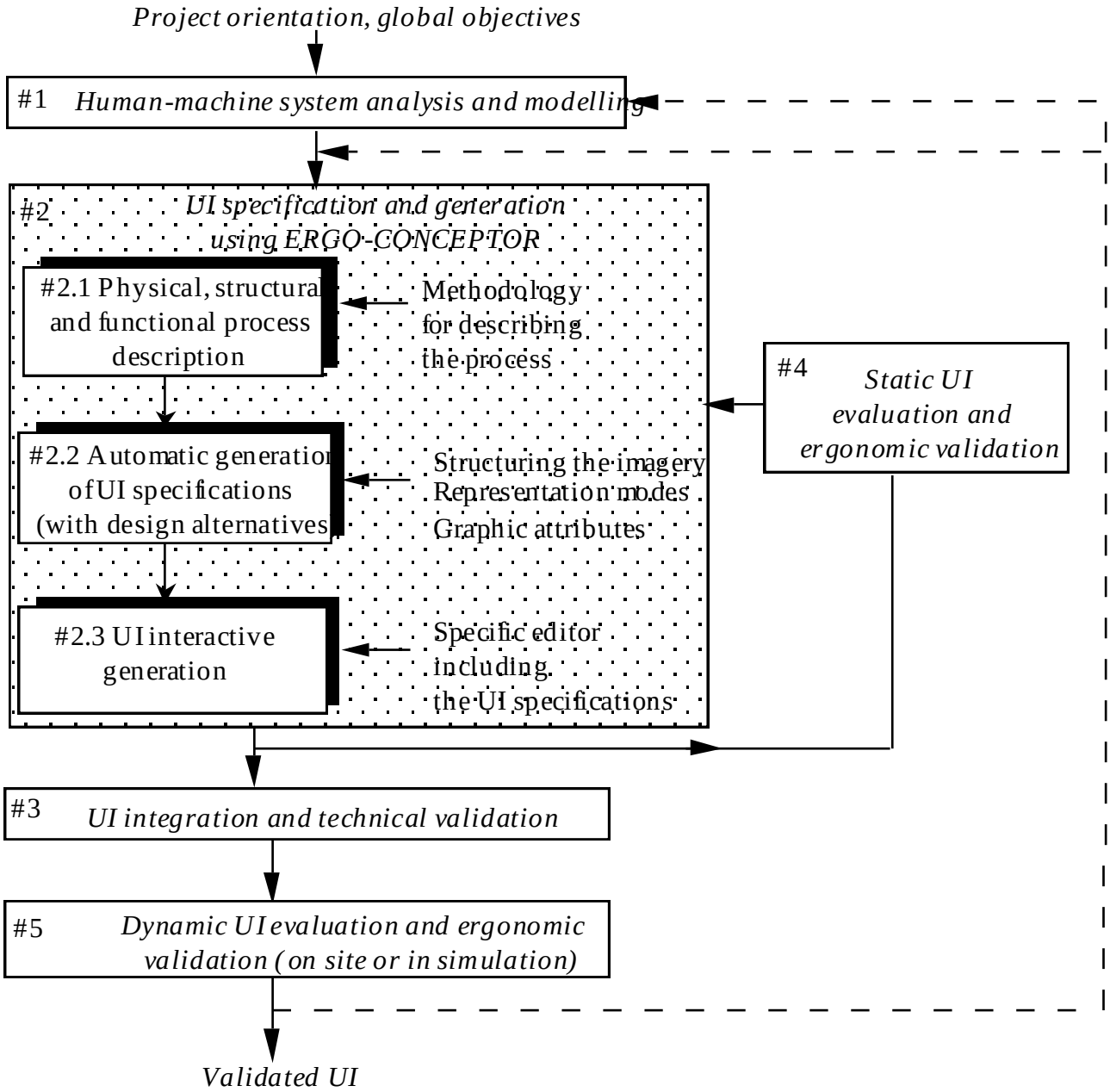


Fig. 2. Global design and evaluation methodology - location of ERGO-CONCEPTOR

In stage #2.3 graphical displays are interactively edited. For that purpose, the designer uses a specific graphical tool. The specifications generated in the second step must be directly usable by the designer through this tool; when design alternatives exist, the human being must be the **final decision-maker**. At the end of this stage, the graphic files are generated and exist on disk.

These three stages are implemented in ERGO-CONCEPTOR by three modules that are used sequentially. In the next section, these three modules are situated in the software architecture of ERGO-CONCEPTOR; they are also progressively explained.

3. ERGO-CONCEPTOR: software architecture and main principles

ERGO-CONCEPTOR is written in Lisp. The version of Lisp used is Le_Lisp version 15.26, associated with the AIDA and MASAÏ graphic environment [61]. AIDA is an object-oriented environment for developing graphic interfaces; this environment integrates an extensible library composed of predefined components: windows, icons, and so on. These components respect the OSF/MOTIF standard. MASAœ is a specific editor able to interactively edit graphic interfaces and then automatically generate files containing AIDA graphic objects. With MASAÏ, each edited object is placed with the mouse on the screen; the attributes (color, size, and character fonts...) can be chosen and modified by the user.

ERGO-CONCEPTOR is made up of the three main modules that are visible in Figure 3. The first allows the description of a set of data relative to the process to be controlled; the process is described in terms of components, systems, sub-systems, etc. The screens allowing the process description have been generated using the MASAÏ editor.

The second module is able to automatically generate UI specifications (eventually composed with design alternatives, see 3.2) with the help of the data described previously. It has been essentially written in LISP.

Finally, the third module has as its goal the interactive generation of a first version of the UI, directly from the specifications that have been automatically generated. These specifications are accessible to the designer (who could be a human factors expert, see Figure 3) through a specialized interface (generated using the MASAÏ editor).

The basic principles used by these three modules are developed in the following three sections.

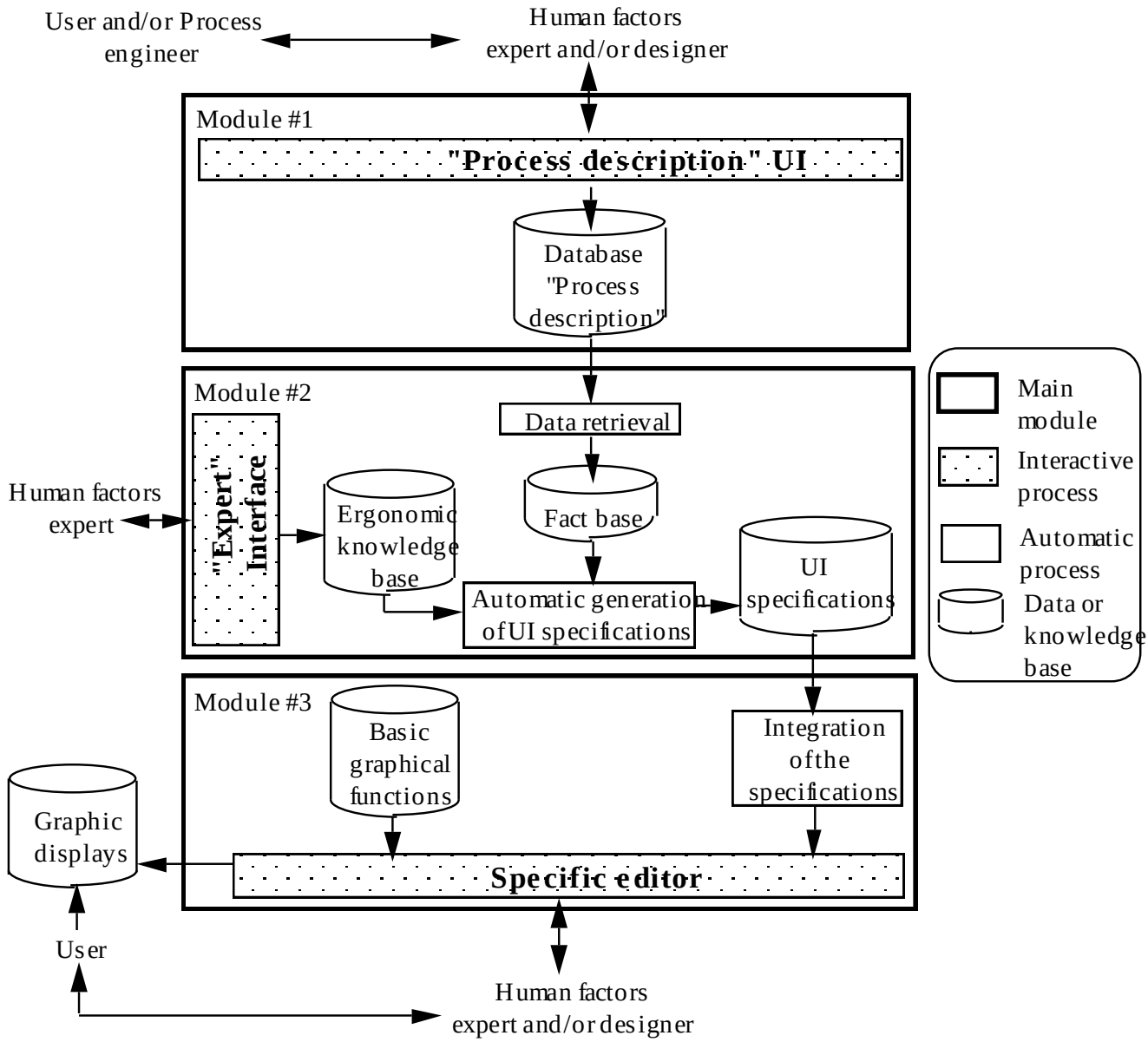


Fig. 3. ERGO-CONCEPTOR software architecture

3.1. Process description (basic principles of the first module)

The only way to control systems, that are complex in terms of large numbers of information sources and devices for basic control actions, is to structure the situation and thereby transfer the problem to a level with less resolution [62]. Different authors, such as Rasmussen [2, 62], Lind [63], Rasmussen et al. [64], Bisantz and Vicente [65], suggest that the structuring can typically be done along two dimensions. One is the whole/part decomposition, in which the system can be seen as a number of related components at several levels of physical aggregation; the second is the level of abstraction in the representation, i.e., the degree to which the physical implementation of functions is maintained in the representation.

In the abstraction hierarchy, the functional properties of a technical system are represented at several levels of functional abstraction along the means-end dimension. This principle is used in our methodology with slight changes needed for the domain of graphical interface generation. In the design of UI for process control, it is very important to design the interface according to two directions :

1. bottom-up: this information is intended to support the operator's identification of the actual state of the process,

2. top-down: the information is needed to help operators in decision-making tasks.

This leads to two problems: to find the information needed, and to structure the representation. To structure the description, a systematic method based on a formal language is required. Lind [63] has approached this problem by developing a multilevel description of process systems in terms of their mass and energy flow topology. This description is basically a representation of the flow topology of the system at several levels of decomposition. The description also maps very well onto a means/end hierarchy for a given system. It is also well suited for structuring a database for graphic support systems independently of the formats chosen for information presentation.

We have used the ideas of both Rasmussen and Lind to provide a formal methodology for process description in the field of the interface generation, particularly for the functional description given below. The process description will ascend the "means/goals" abstraction hierarchy (see below) to allow different users to follow the steps (detection of abnormal situation, evaluation of the process state, decision making and actions) identified by Rasmussen [66] in problem solving. In addition, it allows users to take short cuts according to the different levels of knowledge.

The information expresses progressively a movement from knowledge about process components towards a general and synthetic knowledge (giving by aggregating information about the general purpose of the functions of the system), see Figure 4. So, depending on the abstraction levels, functional groups of variables or sub-systems may be created from other functional groups or from sub-sets of variables (judged as representative, or significant, in accordance with criteria such as importance or pertinence) extracted from the lower abstraction level.

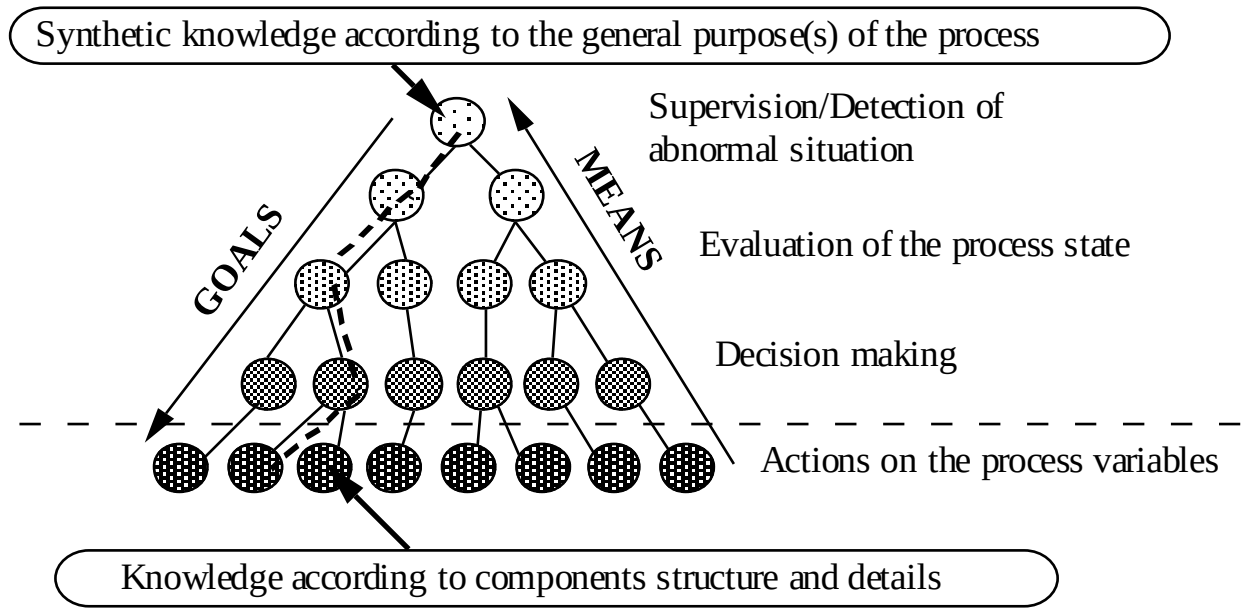


Fig. 4. Pyramid-shape structure of the process description

The "means/goals" abstraction hierarchy principle can be explained as follows : *to allow the user to move among the abstraction levels of the process to solve a given problem, it is supposed that:*

- *at an abstraction level N_i, the goals of a sub-system S_i are reached by acting on the sub-systems of the level N_{i-1} which compose it, and*
- *at the abstraction level N_{i+1}, this sub-system S_i is considered as a means or resource allowing the carrying out of the goals of the sub-system S_{i+1} that involves it.*

So, at the lowest abstraction level, the elements should represent elementary components. They cannot be broken down in the context and for the goals fixed by process experts. At the higher levels of abstraction, the process can be represented, ultimately, by one unique functional group giving a synthetic idea of the whole process [67]. Take, for example, this means/goals description of a simplified process in a nuclear power plant. Three levels of abstraction are considered as being suitable for the control task, figure 5.

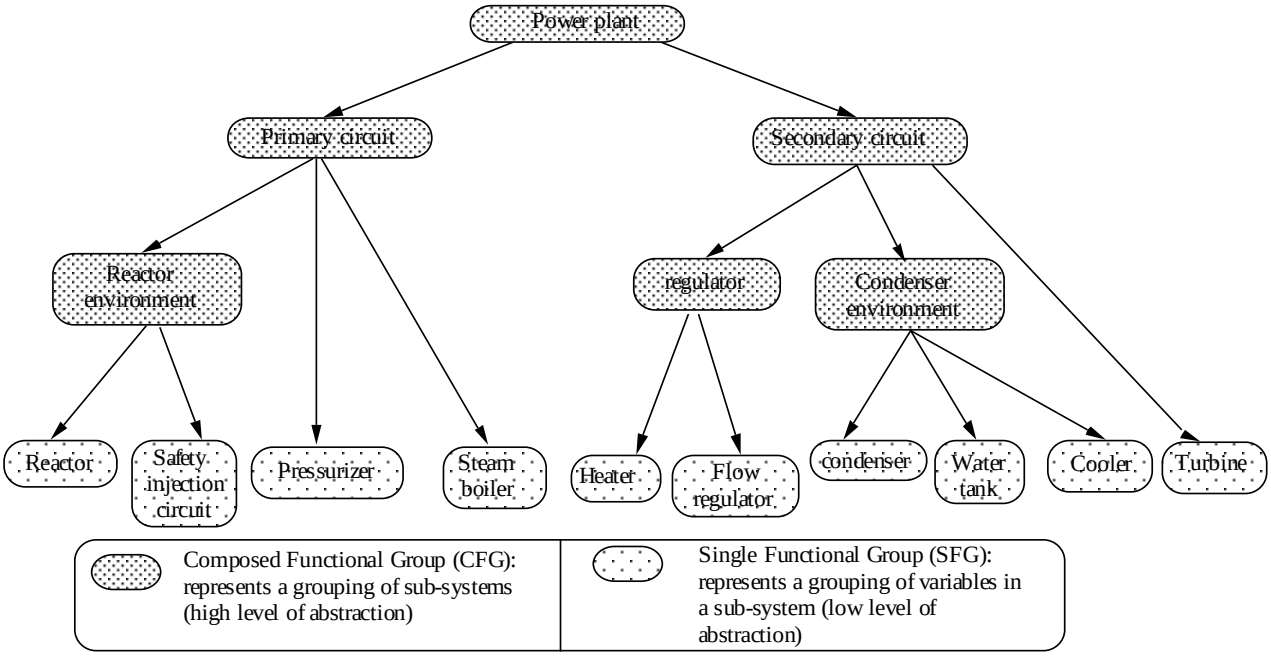


Fig. 5. Example of description in three abstraction levels

The **structural** description is realized by successive regrouping of its sub-systems. [3] At the lowest level, a sub-system is an element that cannot be broken down, represented in Figure 5 by a Simple Functional Group (SFG), involving components connected between them. A Composed Functional Group (CFG) represents a grouping of sub-systems of a higher level of abstraction. The relationships between sub-systems are described by Input/Output links. For example, figure 6 shows successive regroupings and relations between sub-systems described on figure 5.

On figure 6 different variable names appear. For instance, the SFG called "Reactor" contains two variables called DPPR (in French: *Débit de la Pompe Primaire du Réacteur*; translated as 'flow of the Reactor Primary Pump') and PTC (in French: *Puissance thermique du Coeur du Réacteur*; translated as 'Thermic Power of the Reactor Heart'). These names have been given without care of realism by the authors of this paper.

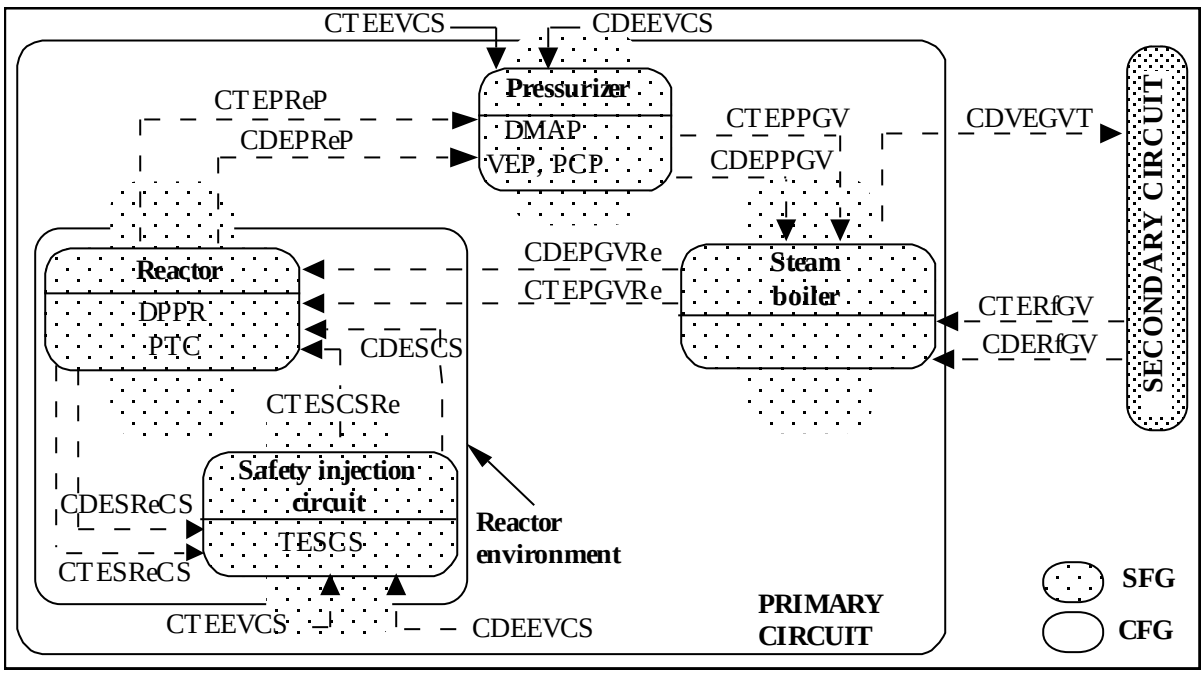


Fig. 6. Simplified example of structural description

[3] In our research, we suppose than the human-machine system analysis and modelling has been made. In particular, according with different authors, such as Goodstein [68], Rasmussen [2], Javaux et al. [69] or Kolski [11], the successive regrouping of sub-systems is really a challenge and requires a collaborative approach involving particularly human operators, designers, production and maintenance engineers, ergonomists. The usable criteria are various: structural or geographic reasons, safety, economy, quality, and so on.

The **functional** description is achieved through the representation of interactions between the different sub-systems that compose the process. It represents interactions by influence links grouped in a causality network. The role of the link is to describe the impact of a variable change on the rest of the process. Tracing the change effect is represented by a network of links called a causality network, as it represents the cause of the change and its origins. The functional description of the process consists in determining, for each Functional Group (FG), the influence links between its variables. The variable upstream, the variable downstream, the propagation direction, the gain, the response time and the delay represent an influence link. It was necessary to analyse the simulated power plant functioning to deduce the list of the links between all the variables. As an example of an influence network, Figure 7 shows the links between variables contained in the SFG called "Pressurizer". The attributes associated with the arrows in the diagram represent (i) the direction of the change that causes a variable of the network on another, (ii) the magnitude of change, and (iii) the propagation delay.

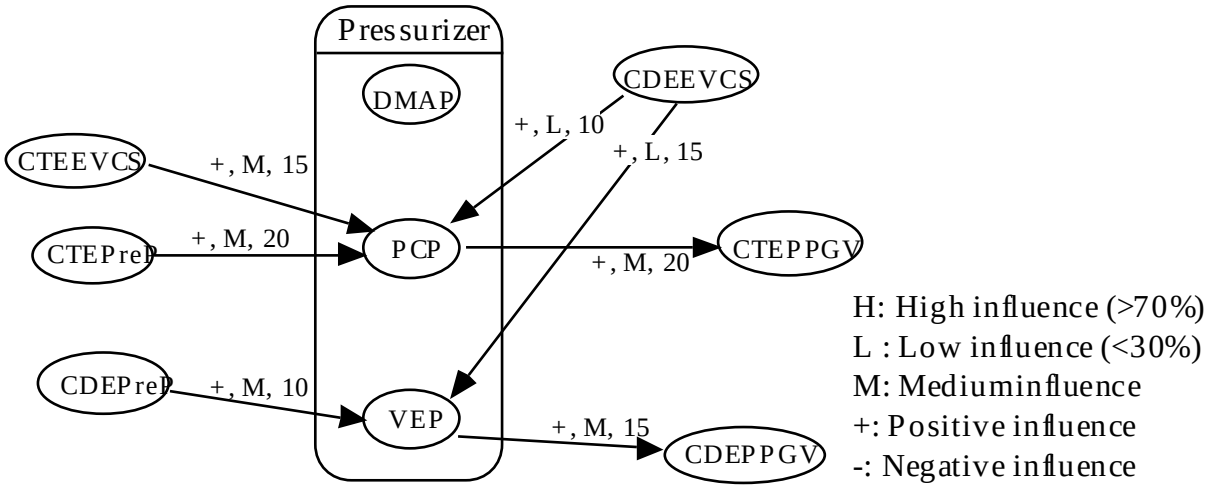


Fig. 7. Simplified example of functional description

Figure 8 illustrates the three descriptions: the means/goals abstraction hierarchy (corresponding to the physical model), the data flow diagram (corresponding to the structural model), and the causality network (corresponding to the functional model). This method of description presents some advantages in the automation of several steps concerning UI design: (1) it uses simple principles, (2) it allows the system representation through different abstraction levels, (3) it presents an iterative aspect, (4) it leads to a database used for the automatic generation of specifications.

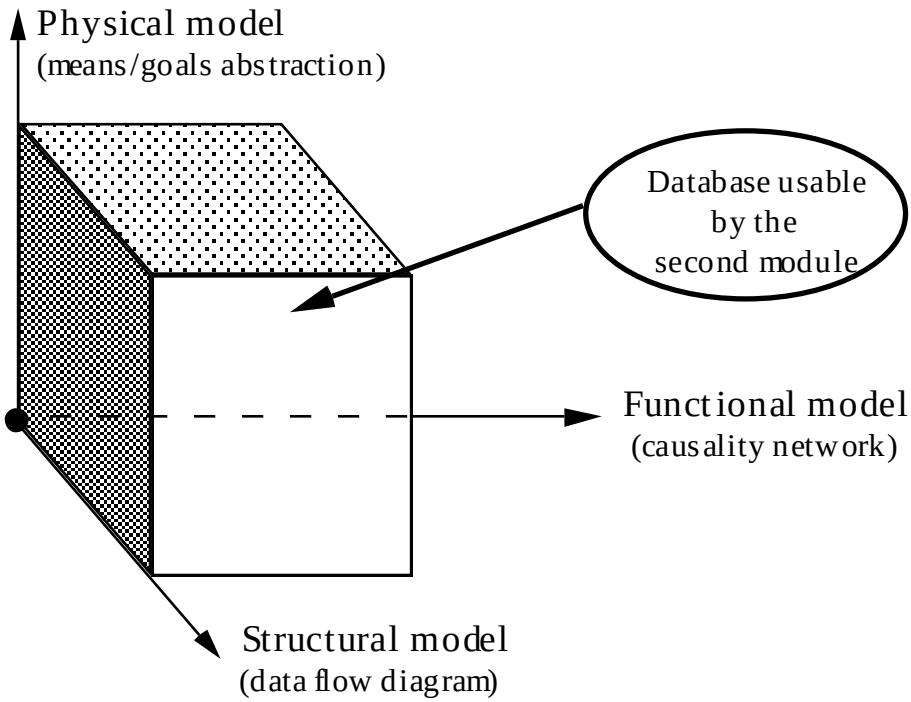
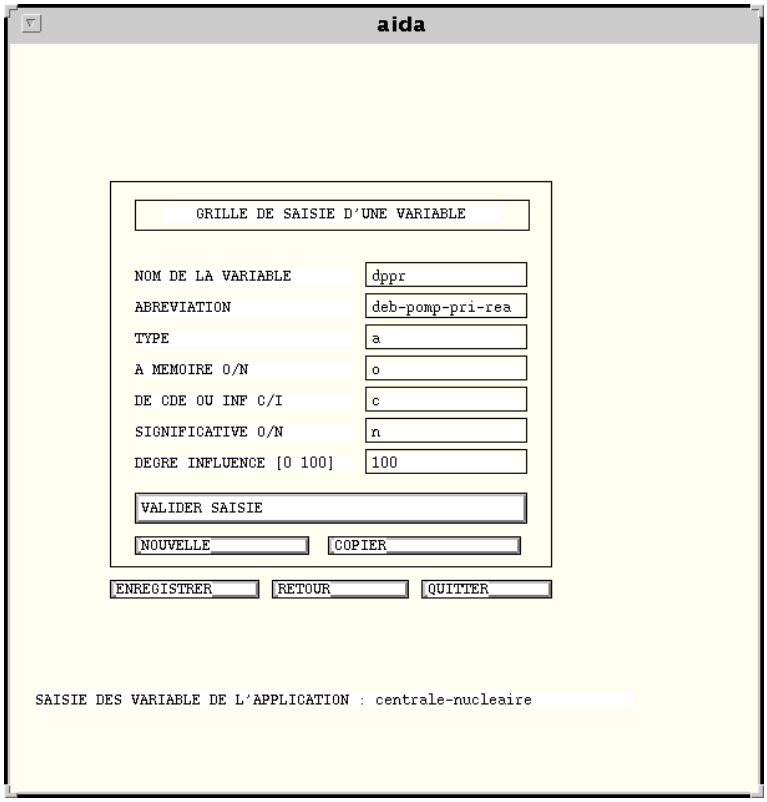


Figure 8. Whole process description (according to the model-based approach terminology)

The first module of ERGO-CONCEPTOR is menu-driven. It is composed of a set of functions for variables, SFG, CFG and causality network management (creation, destruction, saving...). In this version, the process description is achieved by using textual displays. For example, Figure 9 shows a display concerning the textual description of a process variable. Each function is detailed in [70]. The description is stored in a database used for the generation of UI specifications. This database is built with LISP lists. The next module then exploits these data.



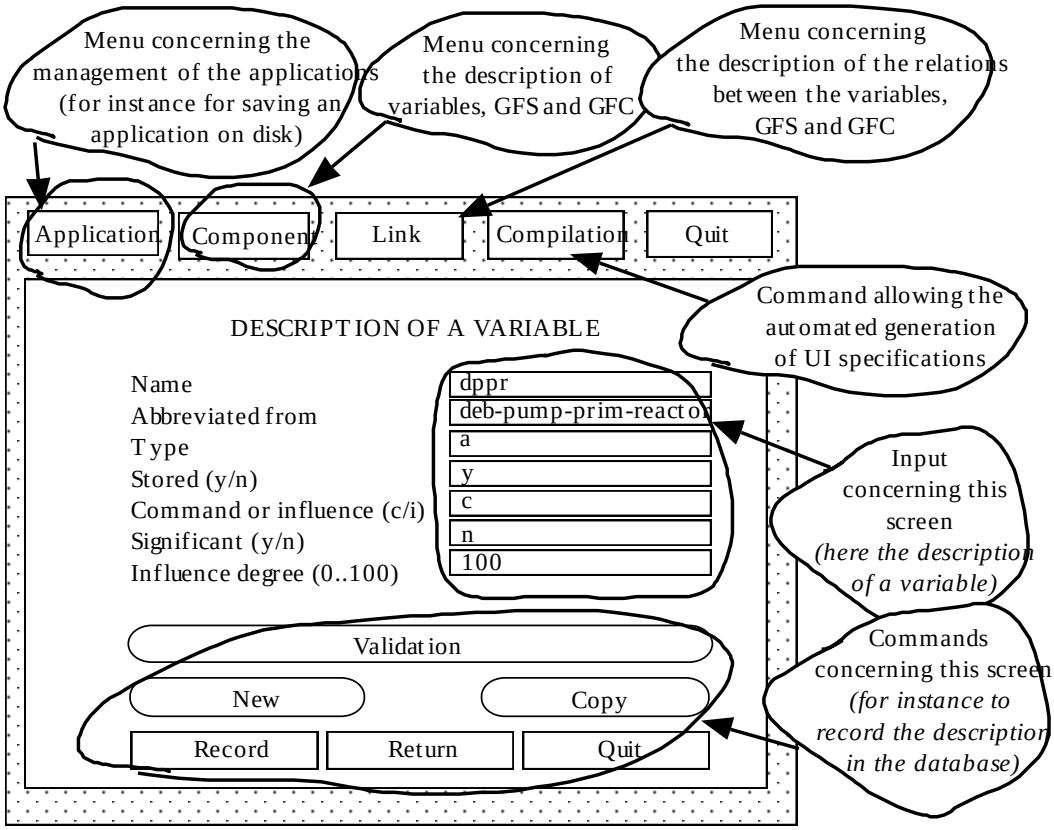


Figure 9. Example of display in the first module : (a) original screen dump of the actual system, (b) translated from French

3.2. The automatic generation of UI specifications

When the process is adequately described, it becomes possible to specify the interface in terms of the SFG and CFG, and then to develop the graphical displays interactively. Currently, only four categories of display are distinguished (more may be considered in future research):

- the "supervision" displays that reflect the process functioning globally,
- the "command" displays that allow the operator to modify the value of process variables,
- the "trend" displays that present the evolution of given variables over time,
- the "causality" displays that present the interconnections between variables (mutual influences) in a SFG.

This list is not exhaustive, but presents a rough idea about the types of graphical displays that could be automatically specified. The “pyramid” model works as follows. At the base of the pyramid are the "command" displays. From the base of the pyramid to its top, "supervision" displays present the sub-systems at different abstraction levels. These displays allow the user to detect abnormal functioning. "Trend" and "causality" displays help the operator during problem- solving steps.

To accomplish this, ERGO-CONCEPTOR exploits a particular approach for the automatic generation of UI specification. On the basis of the data describing the process to be supervised, and with the help of ergonomic knowledge relative to the presentation of the information on the screen, ERGO-CONCEPTOR builds specifications of the complete set of the interface considered as necessary for the process supervision. This module is composed of three sub-modules: one for data retrieval, one is an inference engine, and one for the creation of the specifications (Figure 10).

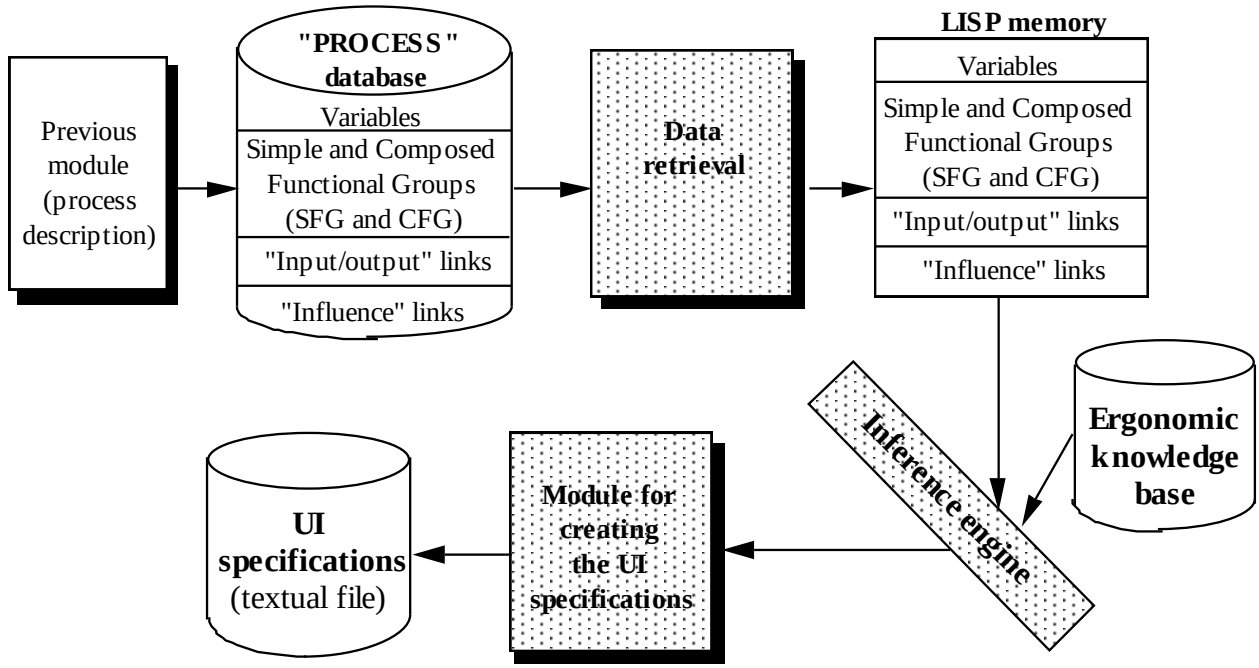


Figure 10. The sub-modules concerning the automatic generation of UI specifications

The sub-module for data retrieval acts as an interface between the first module of process description and the creation of specifications. This sub-module allows the retrieval from the database of all the necessary data. This database is stored in several textual files located in a specific directory (whose name corresponds to the name given to the application). For instance, in the directory called "nuclear", the file called "nuclear.gfs" contains the list of the GFS described in this application (using the first module); each GFS is associated with its intrinsic variables, as follows:

```
***
reactor
ptc
dppr
***
safety-injection-circuit
tescs
***
```

... and so on.

Each file is automatically read; procedures allow the extraction of sub-systems to represent, their inputs and outputs, as well as intrinsic variables and influence links between variables (under the form of a causality network). All data are

stored in the form of LISP lists, which constitute the facts exploited by the inference engine in relation to the ergonomic knowledge base (Figure 10).

The inference engine provides suggested aspects of informational content and graphic presentation to generate. This provides a basic ergonomic assistance for the UI design, and helps to eliminate some risks with errors or incoherence in the resulting UI. The formalisation of ergonomic knowledge consists of rules about information presentation derived from general design guidelines (see for instance Smith and Mosier [71]; Shneiderman [72]; Vanderdonckt [73]) or books specifically oriented towards the process control field [5, 11].

This knowledge is formalised as production rules. Their formalism is the typical one: *If (Condition) Then (Conclusion).* Attributes concerning the previous process description (for instance the presence of a SFG) are located in the condition part. In the conclusion part, one can find types of UI specifications to generate (for instance, a conclusion specifies to create four specific types of view, if a SFG is detected in the condition part). Different examples of rules are presented below. This formalism has been chosen for different reasons:

1. the most important reason is that hundreds of rules already exist in current design guidelines (see upper). It is well known that many rules are contradictory, difficult to understand and/or to use, imprecise, that there is a lack of agreement among experts about ergonomic rules, and so on [74, 75]. Such rules can, however, often be formally represented with condition and conclusion parts;
2. another reason is that important research is focused upon the organization of human factors knowledge for UI evaluation and design, according to ergonomic criteria [76, 77, 78]. Different experiments show that many rules quoted in (1) can be efficiently and usefully structured and regrouped.
3. this formalism has been validated during different projects concerning UI evaluation or design since the eighties. See, for instance, Mackinlay [79], Perlman [80], Kolski et al. [14, 16], Martin [81] or Wiecha et al. [82], where different LISP or PROLOG approaches have been tested, and have proved the feasibility of knowledge-based approaches.

The inference engine exploits these production rules. The mechanism of knowledge processing is based on the classic sequence: defining the total set of applicable rules, choose a rule to apply, fire the inference of the rule, and update. The definition of the total set of applicable rules is obtained by verifying successively for each rule whether the condition expressed in the left side of the rule is satisfied. The inference of the rule allows the deduction of whether the conclusion expressed on the right hand side is satisfied. The inference engine is used by the sub-module that tries to create the specifications.

This sub-module aims to create the specifications and writes to a file containing the specifications in a textual form. The specifications are expressed in a predefined syntax based on a grammar (inspired by Olsen [46]; for details see [70]). This syntax represents a formal entity definition of all the abstract interaction objects that will be used during the later interactive generation of the human-computer interface (the approach is globally the same as those of Puerta [53]). A view, for example, will be defined by a descriptor called "identifier-view" and by a list of zones "description-zones". Each zone is then able to be then more precisely described, according to the principle shown in Figure 11. This syntax has two advantages:

1. It can define *a priori* a set of abstract interaction objects, from the more simple (graphic attributes: colors, character sizes...) to the more complex (type of views, structure of views, presentation modes, etc) which can be altered.
2. It can ensure the independence of the module for the description of the process and the graphic editor that needs to be modified or replaced. The recourse to a known syntax allows the avoidance of all incoherence problems between the computer data structure processed by the generator of specification and the specific format of data processed by the graphic publisher chosen. Furthermore, the use of formal grammar forces one to take note of the graphic modes predefined *a priori* or commonly used in the application being considered.

3.

```
<SPECIFICATIONS> -> <VIEW>
<SPECIFICATIONS> -> <VIEW> <SPECIFICATIONS>

<VIEW>  -> <IDENTIFICATION-VIEW> <DESCRIPTION-ZONES> "FIN ZONE"

<IDENTIFICATION-VIEW> -> "VIEW" <NAME-SUB-SYSTEM> <TYPE>

<DESCRIPTION-ZONES> -> <ZONE>
<DESCRIPTION-ZONES> -> <ZONE> <DESCRIPTION-ZONES>

<NAME-SUB-SYSTEM> -> <IDENTIFIER>

<IDENTIFIER> -> <ALPHABETIC-NAME> <NUMBER-INTEGER>

<ALPHABETIC-NAME> -> <CHARACTER>
<ALPHABETIC-NAME> -> <CHARACTER> <ALPHABETIC-NAME>

<CHARACTER> -> a / ... / z / A / ... / Z

<NUMBER-INTEGER> -> <INTEGER>
<NUMBER-INTEGER> -> <INTEGER> <NUMBER-INTEGER>

<INTEGER> -> 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9

<TYPE> -> "SUPERVISION" / "COMMAND" / "CAUSALITY NETWORK" / "TREND"

<ZONE> -> "ZONE" <NAME-ZONE>

and so on.
```

Figure 11. Some examples of the grammar used to generate the specifications file.

The content of the file thus obtained describes for each view: its type (supervision, command, trend...), the sub-system, zones that the view includes (structural definition) and each zone content (informational entities). Each zone presents a set of informational entities. Specific rules may be used to choose the graphical attributes and the representation modes used to represent these entities. A set of production rules can cover the necessary decisions.

Some simplified examples of rules derived from the knowledge base include:

IF CFG
THEN supervision-view AND causality-view AND trend-view

IF supervision-view AND intrinsic-variables and number-less-than-10
THEN star-view

IF view-to-create
THEN title-zone AND alarm-zone AND work-zone and navigation-zone

IF dangerous-state
THEN red-color

The first rule states that each CFG will need to create supervision, causality and trend views. The second means that if one is in the process of creating a supervision view and that this view will comprise less than 10 intrinsic variables, then the graphic mode "star" is possible. The third specifies that each view being created could be composed of the following zones: title, work, alarm and navigation zones. Finally, the fourth rule states that a state of danger will be associated with the color red.

As is easily seen in these examples, knowledge contained in the ERGO-CONCEPTOR base includes the structuring of views, the graphic attribute choice (colors, character font...) and presentation modes of representation (star, bar graph...). The type of ergonomic knowledge is inferred from the level of abstraction of the process of view construction. The main steps of the global approach for creating specifications of the interface are resumed in figure 12: each set of identifications is taken into account by specific rules studying the content of the LISP lists. Each inference leads to updating a global LISP list containing the UI specifications.

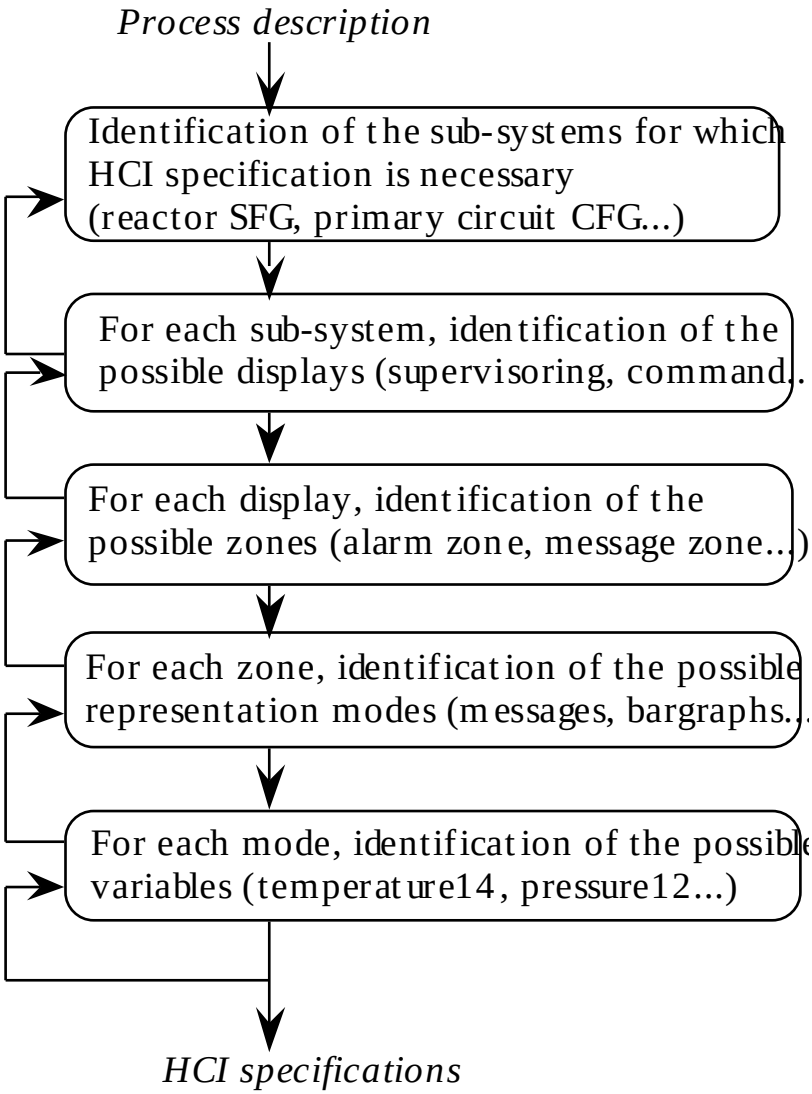


Figure 12. Steps of the algorithm used for the creation of the specifications

According to this algorithm, when constructing a view, it is necessary to first determine the required structure for the view in the context of the application. Next, at a lower level abstraction (construction of the zone of work, for example), one chooses in the same way which necessary set of variables to display. Finally, at the lowest level of abstraction, it specifies which graphic attributes to use.

The result of this stage of specification generation is a text file written in a special language [70]. A short extract from this file is shown in Figure 13. This file can be exploited directly by the designer as paper documentation. Nevertheless, its main use is to directly support the interactive creation of graphic views, which feature in the next sub-section.

```
*****************************************************************
                TYPE OF VIEW                          : command
                NAME OF THE SUB-SYSTEM                : reactor
                TYPE OF THE CORRESPONDING FG : simple
*****************************************************************
 BEGIN VIEW
   ZONE: Title
     ENTITY: rectangle
       Position: 80 , 115
       Dimensions: 850 , 40
       Background color: blue
       Foreground color: black
       Line type: continue-3
     END ENTITY
   ...
   END ZONE

...

   ZONE: Work
      ...
     ENTITY: Variables concerning the commands of the reactor sub-system
       Variable name: ptc
       Variable name: dppr
     END ENTITY
      ...
   END ZONE
...
```

Fig. 13. Short extract of a specification file automatically generated (translated from french)

3.3. Principles of interactive creation of the UI, based upon the use of the previously generated specifications (third module of ERGO-CONCEPTOR)

The graphic view creation uses specifications automatically generated by the preceding module. These specifications describe the complete set of graphic views to be created, with all details concerning their structure, the composition of the different zones and the graphic attributes to be implemented. A graphic view editor, at high level of abstraction, is proposed as an aid to the designer. This editor can be called at high level of abstraction because the interactive creation is based on intermediate specifications at different levels. The reason for this is explained above, from the highest level (the structuring of the imagery as a function of sub-systems composing the process), to the lowest level, graphic attributes compose each view.

When editing graphics, designers have be able to exploit directly the specifications which have been generated in order to justified modifications to views. Designers must also be able to reject specifications and to use a more personalized graphic creation style. In this way the maximum degree of liberty is left to the designers to avoid constraints common to many rigid editors. In others words, designers will be assisted "intelligently" during graphic view creation, rather than being directed. This is a fundamental principle of the ERGO-CONCEPTOR system.

This module passes the UI specifications to the designer. Basic graphical functions are also available for drawing a rectangle, a circle, for editing a text, and so on. One must realise that there are no particular relationships between the UI specifications and the graphical functions. Figure 14 shows the user interface of this interactive module. The specifications are reachable with the menus called "sub-systems", "type of display", "zone to create", "representational modes" and "list of variables". The basic graphical functions are available on the right side of the editor.

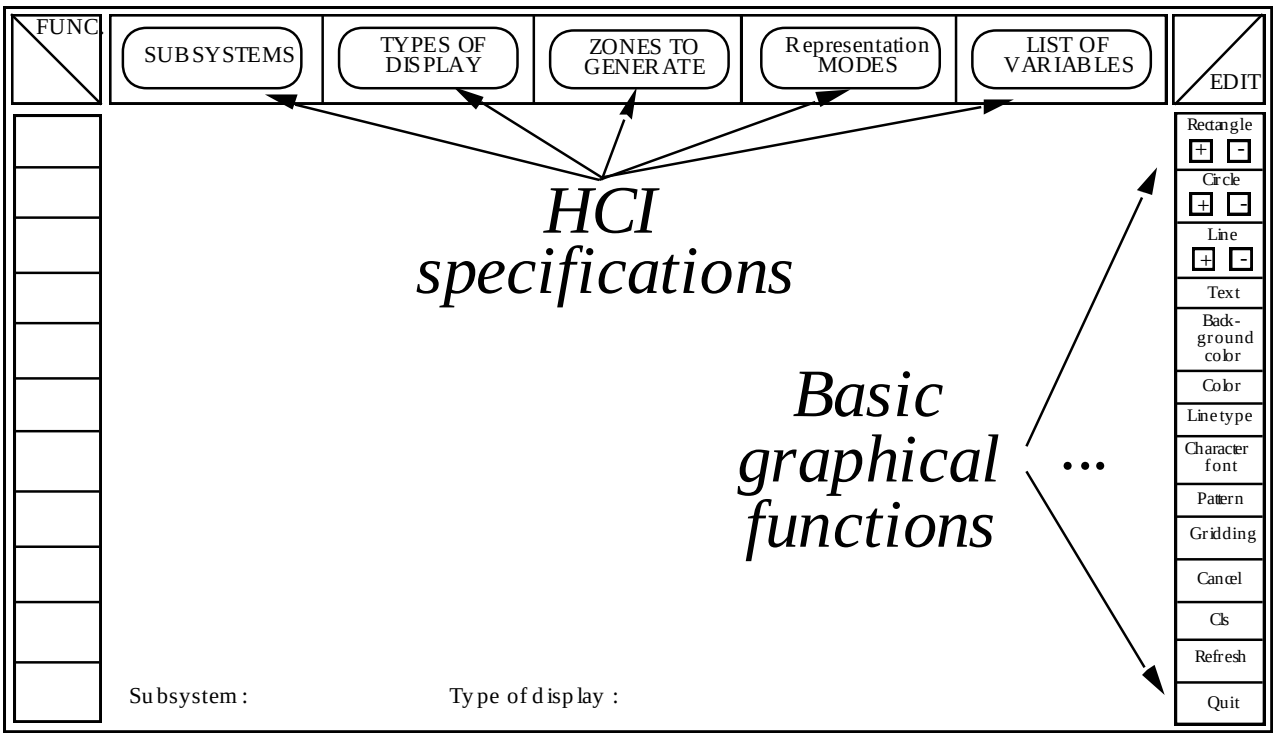


Figure 14. Interface structure for the creation of graphical displays (translated from French)

In the next section a first validation of ERGO-CONCEPTOR is explained with examples of edited displays.

4. Initial technical validation of ERGO-CONCEPTOR

A technical validation has been made based on diagrams and explanations of a power plant (see for instance [83]). This process involves a typical steam generation process. Fuel burns in a boiler, and yields its energy in the form of heat to water, which will be vaporised as steam. The steam is used to drive a turbine, which is coupled to an alternator to produce electricity. This process is composed of a "primary" circuit and a "secondary" circuit. The primary circuit can be conceptually and functionally broken down into four sub-systems: reactor, safety injection circuit, pressurizer, and steam generator. The secondary circuit can be broken down into six sub-systems: turbine, water tank, cooler, debit regulator, heater and condenser.[4]

The technical validation stage has led to the interactive generation of more than 50 displays. An example of generated display is shown in figure 18. This display is associated with the "environment-condensor" sub-system. It is of the "supervision" type (Cf. III.2), which allow the user to supervise the different sub-systems of the process. So, such a display presents the state of a given sub-system by displaying the current values of representative variables. The mode of information presentation selected by the designer is the "star" [59, 84]. The designer has decided to show five variables using this representation mode (cdeprep, cdepgvre, ptc, ctepgvre, dppr). The main steps followed by the designer are explained in the following sub-sections.

[4] Important remark: the human-machine system analysis and modelling phase has been simulated in our laboratory (for more details see [70]). It has not been performed by process engineers and ergonomists, so its results do not reflect complete reality. Since this step is not important for our research, the assumption was made that this simulation provides us with all the data necessary for the first module inputs. According to this hypothesis, the technical validation was possible.

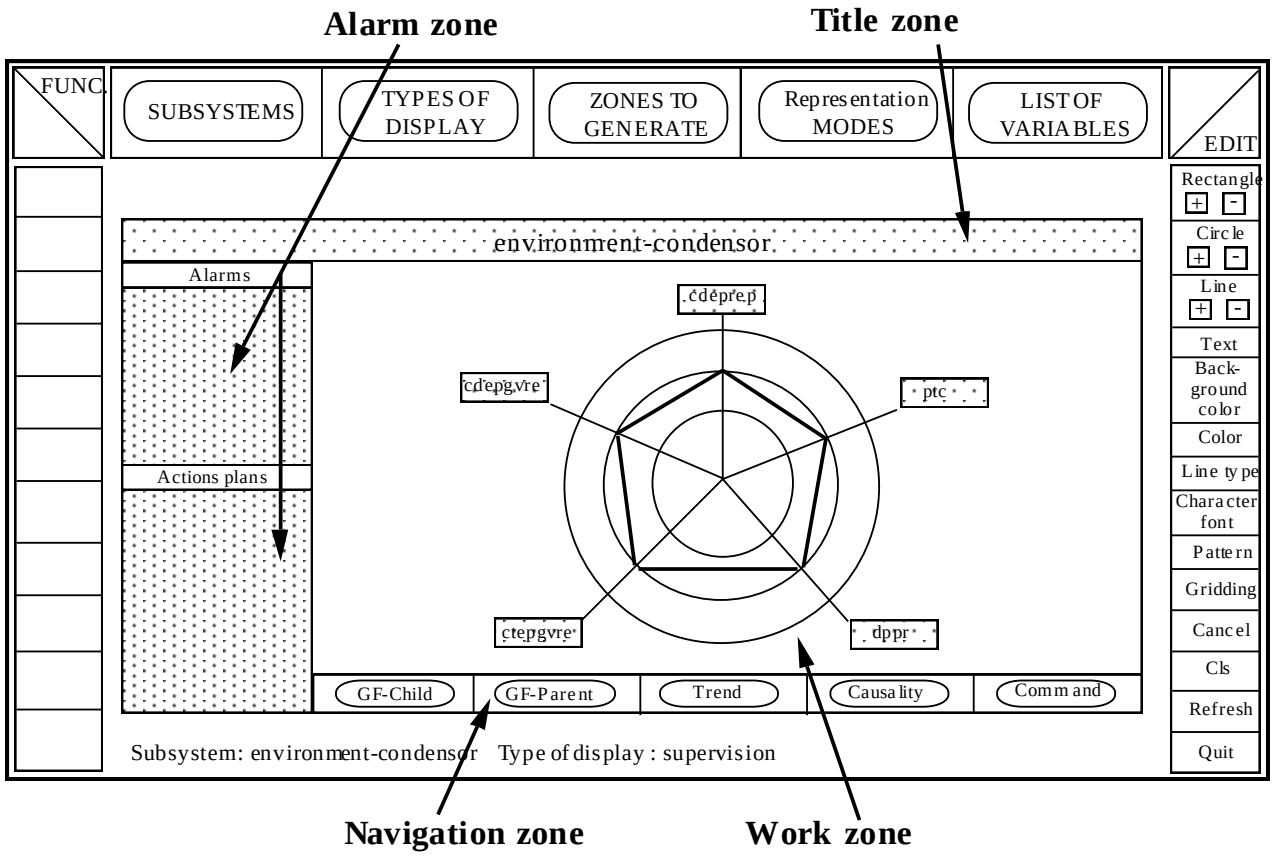


Figure 15. An example of generated display using the "star" mode (translated from French)

4.1. The steps followed by the designer using the first module

Using the first module, each of the main components of the power plant can be described by a SFG. Thus ten SFG are created, namely reactor, safety injection circuit, pressurizer, steam generator, turbine, water tank, cooler, debit regulator, heater and condenser. These SFG have been grouped in more global CFG, (as in Figure 5). Moussa [70] presents the complete description of all the variables of each SFG and CFG. Figure 5 summarises the hierarchical description of the process, using abstraction levels. In this tree, a node represents a CFG; a leaf represents a SFG. Each level in the tree means a change of abstraction level giving different points of view of the power plant. In fact, the power plant is considered as an arborescence where the root called "power plant" reflects a synthetic knowledge relative to the general purpose of the process, and where the leaves (SFG) represent a deep knowledge relative to the structure and to the details of the components.

The functional description of the process consists in determining, for each FG, the influence links between its variables (according to the principles previously described, Figure 7).

When the process is initiated by using the first module, the UI specifications generation is activated with the button "Compilation" (see the top level of the diagram in Figure 9). The specification generation is completed in two to three minutes, after which the UI specifications exist in LISP memory. The interactive creation of the graphical displays using the deduced UI specifications becomes possible. The UI specifications are integrated into the interactive editor (corresponding to the third module of ERGO-CONCEPTOR), and can then be used by the designer.

In Section 3.2, the main principles for the UI specifications generation were described. In the next section, the different steps followed by the designer, to interactively edit the displays, are explained.

4.2. The steps followed by the designer using the third module

The designer first selects a sub-system among those proposed. Each sub-system belongs to the list of the SFG and CFG previously identified (by the second module). This is accomplished by activating the button called "SUB-SYSTEMS" and entering, in the string selector, the name of a sub-system. The displays of this sub-system have then to be edited (Figure 16). For instance, for the display showed in figure 15, the "ENVIRONMENT-CONDENSOR" sub-system has been selected.

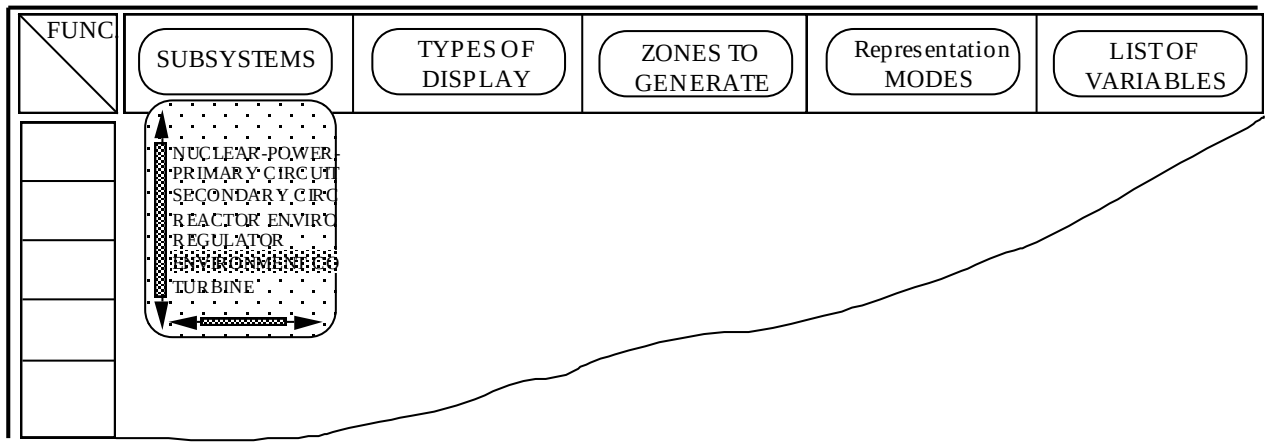


Figure 16. Selection by the designer of a sub-system (translated from French)

When the sub-system is chosen, the designer selects the type of display to generate. This is done by selecting the button entitled "TYPES OF DISPLAYS" (figure 17) ; the designer selects, among the list of types proposed, the required type. In this version of ERGO-CONCEPTOR (and as explained in section III.2), this list can be composed from displays of four types: "supervision", "causality" and "trend" for the CFG; for the SFG, the "command" type is added. For instance, for the display showed in figure 15, the "SUPERVISION" type has been selected.

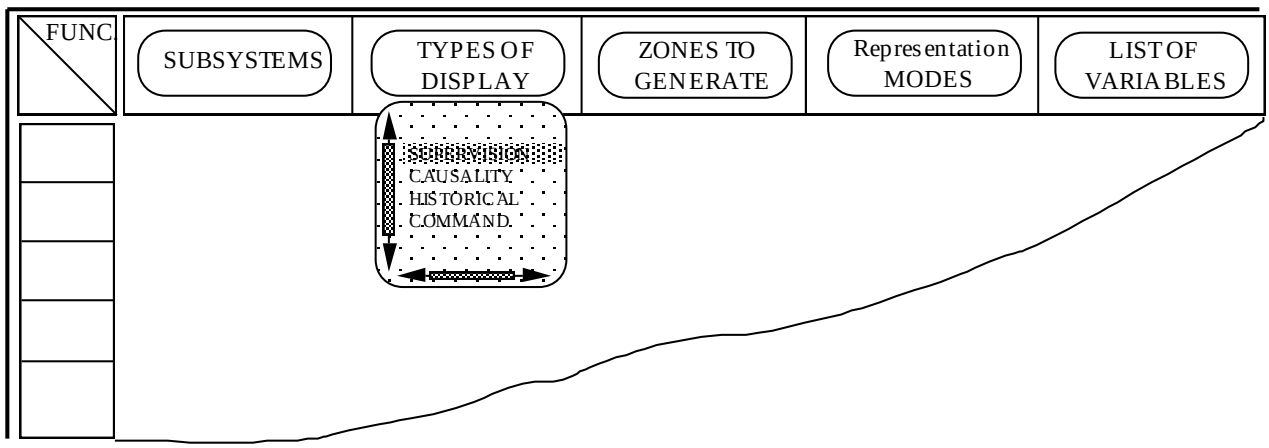


Figure 17. Selection of a type of display (translated from French)

Once the sub-system and its type are known, different zones composing the display to generate are suggested to the designer. When the button called "ZONES TO GENERATE" is selected, the zones to edit can be chosen successively in the string selector. When creating the "work zone" the designer chooses a graphical representation mode among those selected by the second module and which corresponds to the features of the given display.

Thus, the designer activates the button entitled "REPRESENTATION MODES" then enters, in the string selector, the required mode. The second module deduced these modes. Some examples of edited modes are given in section 4.2. For instance, for the display showed in figure 15, the "STAR" mode has been selected (the other possibility was the "BARGRAPH" mode).

Finally, the designer chooses the variables to be represented, by activating the button called "LIST OF VARIABLES" and enters, in the string selector, those variables that are of interest. The second module also deduced these variables. For instance, for the display showed in figure 15, five variables have been selected (cdeprep, cdepgvre, ptc, ctepgvre, dppr).

Throughout these steps, the designer can use the basic graphical functions to perform modifications such as changing colors, size, shape, and so on (Figure 14).

4.3. The structure of the generated displays

In this application, and in accordance with the ergonomic knowledge used in the second module, the generated displays of the application "power plant" are composed of four zones (figure 15):

- The title zone. This involves the identification of the sub-system.
- The work zone. This involves the variables that characterise the state of the current system. These variables are displayed according to the representation modes selected by the designer.
- The alarm zone. This zone is intended for a connection between the displays and an alarm management system.

- The "navigation" zone. This allows the user to "move" along the displays' tree and to shift the type of display (supervision, trend and so on) associated with a given sub-system. An example of "navigation" zone is displayed in Figure 18. This zone involves (i) a button called "FG-PARENT" to go to the sub-system parent of the current sub-system, (ii) a button called "FG-CHILD" to go to a chosen sub-system among the children of the current one. There are also several buttons giving an access to the other possible displays for the current sub-system.

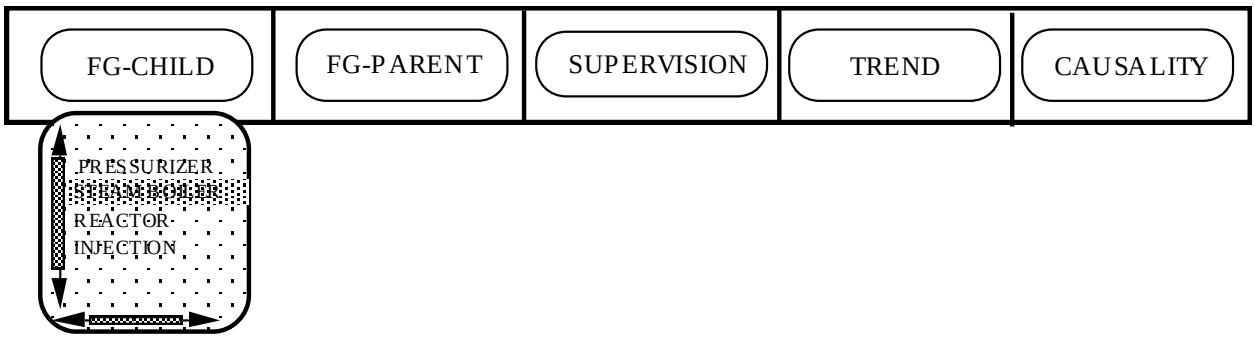


Figure 18. Example of "navigation" zone (translated from French)

4.4. Other examples of generated displays

Each functional group is considered as a sub-system associated with a set of displays. Thus, for a CFG, "supervision", "trend" and "causality" displays are proposed in the UI specifications and then interactively edited by the designer. For a SFG, in addition to these three displays, "command" displays can be generated. Two other examples of generated displays are presented below.

- The "causality" display presents the influence links between the different variables of a sub-system, and the links with other sub-systems. The graphical mode chosen for this kind of display is called "causality network" (Figure 19).

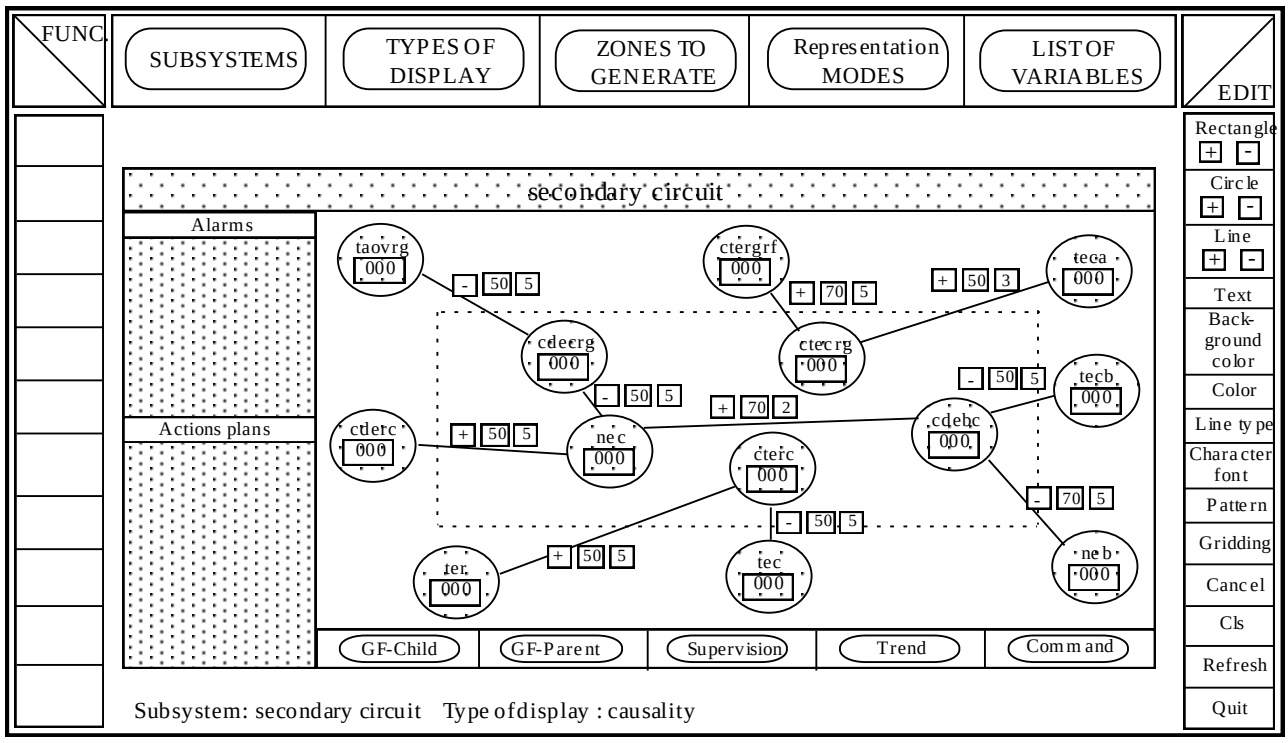


Figure 19. An example of edited display using the "causality network" mode (translated from French)

- The "trend" display presents the past values of variables using curves. The graphical mode of this display is called "historical network". Figure 20 shows such a display with (i) in the center of it the evolution in time of a selected variable and (ii) the evolution of the variables in relation to it.

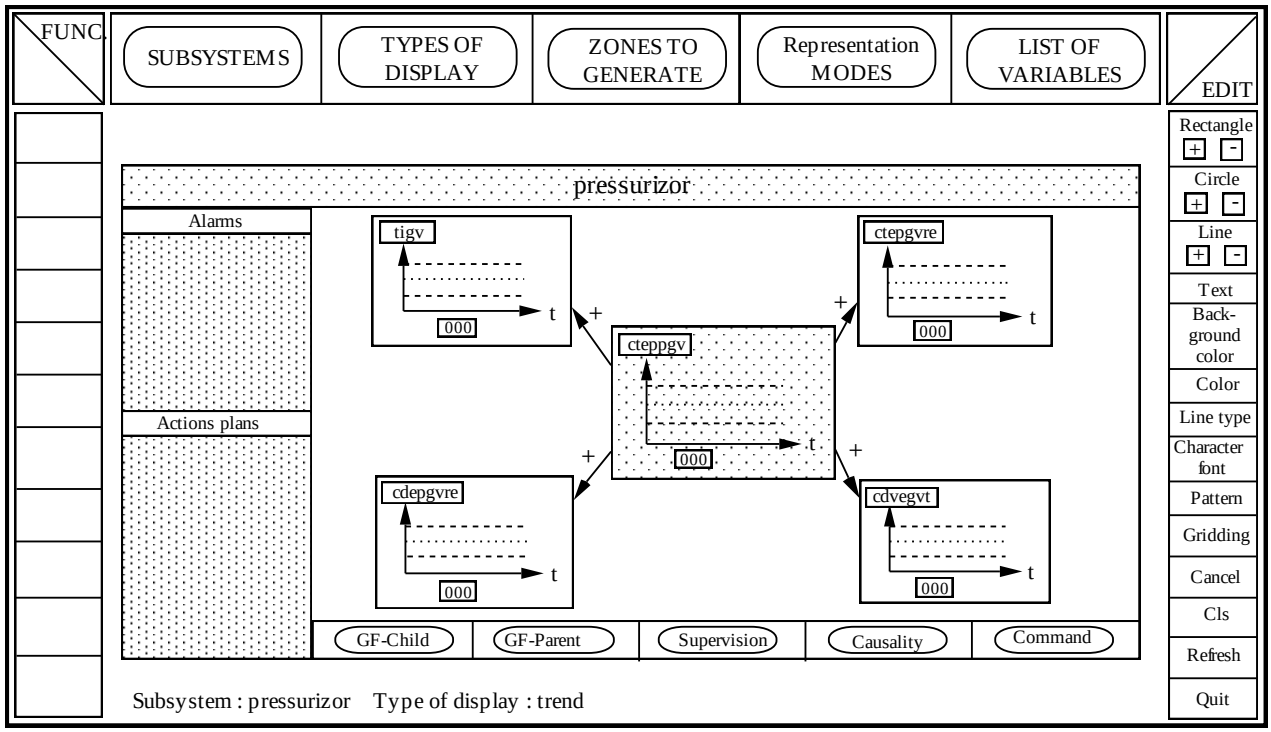


Figure 20. An example of edited display using the "trend" mode (translated from French)

5. Conclusions

This article has proposed a model-based method for UI specification. This method is implemented as a software tool called ERGO-CONCEPTOR. Its feasibility has been demonstrated in a first validation carried out in our laboratory: *From the description of an application, it is possible to automatically generate UI specifications and to provide a designer with these specifications to use as guides during the design task.*

ERGO-CONCEPTOR is, at present, an experimental tool. Its three modules need to be considerably improved. For instance, an ergonomic evaluation of ERGO-CONCEPTOR carried out in our laboratory with potential users (engineering students) showed that the first module is difficult to use because of the textual screen pages. For the next version of ERGO-CONCEPTOR, we intend to implement a graphical interface for this module. Furthermore, the ergonomic knowledge contained in the second module is not exhaustive; the knowledge base needs to be progressively extended. Nevertheless, the results of the first validation are very encouraging.

In spite of its current ergonomic and technical shortcomings, the ERGO-CONCEPTOR approach is promising. It seems that future tools based on such principles could be useful in many situations. Indeed, in an application field where the ergonomic rules are clearly defined and validated by experts, for certain low level aspects, the UI design tasks may become *repetitive* [85] rather than time consuming, expensive processes involving original creativity on every occasion.

Without discussing the adequacy of the displays for real user requirements (see our hypothesis, section II), one notices, in the process control field, that *only a limited number of representation modes, symbols and graphical attributes are used in displays found in control rooms*. Adopting a knowledge-based approach to generate UI specifications is then possible. Of course, the validation of such knowledge will be progressive and it will be a long time before the integration of this knowledge into future systems based on the ERGO-CONCEPTOR principles is complete. One can conclude that real advantages of incorporating guidelines in ERGO-CONCEPTOR are:

- The designer has not to rebuild himself or herself a mental representation of the industrial process nor ask himself or herself which sub-systems need an UI, since some of the guidelines undertake the identification of the relevant sub-systems, at different abstraction levels.
- For each sub-system, adapted types of displays are automatically proposed to the designer (supervision displays, command displays...). Thus, some of the guidelines avoid the designer to forget important types of displays, for instance, a display for commanding the water-gates of a sub-system.
- For each type of display, it is well known that an important problem consists of proposing a screen format, constituted with different zones and/or windows [86]. The systematic use of guidelines concerning these zones allow to simplify the design task by proposing adapted formats.
- In process control, predefined representation modes exist (star mode, bargraph...). It is important to use them deliberately for each type of display. Including guidelines in a design approach for their selection avoids numerous ergonomic errors. Similarly and based on this idea, the choice of certain graphic attributes which are important in process control (such as the colors) can be also very fruitful. For instance, a common design error consists in using the red color without any relation with alarms or dangerous situations.

For the two first points, we wish to insist on the fact than these guidelines become efficient only if they are connected with a model describing the chosen application. Then, the advantage of a model-based development approach becomes very important (in our particular case, with an approach using physical, structural and functional models, see II.2.3). Usually, without assistance, all these four points lead to a difficult and tedious work for the designer during the specification stage. Our first validation has been proved fruitful, even if the knowledge base is incomplete, because (1) these points can be traited using a chaining of (previously formalised) guidelines, (2) our approach leads fastly to the generation of UI specifications. The time profit can then be exploited by the designer for evaluating, improving and validating these specifications with the human operators, before the effective UI realization. Another advantage of a model-based development approach is to incite the designer to analyse and model in depth the human-machine system.

Two other fundamental ideas can also be expressed. These ideas are subject of research perspectives:

- *Concerning the possible modification of the user requirements.* Sometimes, when the process is submitted to physical transformations, or more generally, when new user requirements appear, it is necessary to modify the information presented to users. Usually the designer has then to modify displays. Design errors and omissions are possible. The advantages of application description principles like those used in the ERGO-CONCEPTOR's second module then become clear. The necessary work for the designer consists simply in modifying the process description (in the form of a database). Then UI specifications may be again automatically generated. The need for upgrading, or completely redesigning, the displays involved would be certainly better identified.
- *Concerning the modification of the ergonomic knowledge base.* When a tool such as ERGO-CONCEPTOR is used, it is possible to allow experts (in human-computer communication) to operate and act at a more conceptual level than during many current projects. If new knowledge about information presentation on a screen appears and is formalised and integrated in the knowledge base, it is possible to take it automatically and systematically into account during the UI specification stage.

Another research perspective concerns the modelling of the UI dialogue using interpreted Petri nets [87].

Currently there are many researchers at work on model-based development tools such as the members of the Special Interest Group "Tools for Working with Guidelines" of the ACM SIGCHI [88]. We think that our approach could influence a new generation of model-based development tools in process control domain.

Acknowledgements

The authors sincerely thank Professor Jean Vanderdonckt (editor-in-chief of this series) and the anonymous referees of the journal for their numerous helpful comments. They thank also Joe Galway and Professor Neville Moray for their considerable assistance in the translation of the article.

References

[1] U.S. Nuclear Regulatory Commission (1981). *Guidelines for control room design reviews* (NUREG-0700). Washington, D.C. : U.S. Nuclear Regulatory Commission.

[2] Rasmussen J. (1986). *Information processing and Human-Machine interaction, an approach to cognitive engineering*. Elsevier Science Publishing.

[3] Millot P. (1988). *Supervision des procédés automatisés et ergonomie*. Editions Hermès, Paris.

[4] Sheridan T.B. (1988). Task allocation and supervisory control. In *Handbook of Human-Computer Interaction*, M. Helander (ed.), Elsevier Science Publishers B.V., North-Holland.

[5] Gilmore W.E., Gertman D.I. & Blackman H.S. (1989). *User-computer interface in process control, a Human Factors Engineering Handbook*, Academic Press.

[6] Johannsen G. (1992). Towards a new quality of automation in complex man-machine systems. *Automatica*, vol. 28 (2), pp. 355-373.

[7] De Keyser V. et al. (1992), *The Nature of Human Expertise.* Intermediary report, convention RFO/AI/18, University of Liège, Faculté de Psychologie et des sciences de l'Education, Belgium.

[8] O'Hara J., Stubler W., Brown W., Wachtel J. & Persenky J. (1995a). Comprehensive guidance for the evaluation of human-systems interfaces in complex systems. *Proceedings of the Human Factors and Ergonomics Society, 39th Annual Meeting*, San Diego, California, USA, October 9-13.

[9] O'Hara J., Brown W., Stubler W., Wachtel J. & Persenky J. (1995b). *Human-system interface design review guideline* (Draft NUREG-0700, Rev. 1). Washington, D.C.: U.S. Nuclear Regulatory Commission.

[10] Hoc J.M. (1996). *Supervision et contrôle de processus, la cognition en situation dynamique*. Grenoble : Presses Universitaires de Grenoble.

[11] Kolski C. (1997). *Interfaces homme-machine, application aux systèmes industriels complexes*. Editions Hermès, Paris.

[12] Elzer P., Borchers H.W., Weizang C. & Zinzer K. (1988). Knowledge-supported generation of control room pictures. *Proceedings IFAC Workshop Artificial Intelligence in real-time control*, Clyne Castle, Swansea, Great Britain, September.

[13] Elzer P. & Johannsen G. (1988). *Concepts, design and prototype implementations for an intelligent graphical editor (IGE1)*. ESPRIT-GRADIENT P857, Report N° 6, Laboratory Man-Machine Systems, University of Kassel, Germany.

[14] Kolski C., Van Daele A., Millot P. & De Keyser V. (1988). Towards an intelligent editor of industrial control views, using rules for ergonomic design. *IFAC Workshop Artificial intelligence in real-time control*, Clyne Castle, Swansea, Great Britain, 21-23 September.

[15] Johannsen G. (1990). Design issues of graphics and knowledge support in supervisory control systems. In W.R. Ferrell, W.B. Rouse and N. Moray (Eds.), *Robotics, Control and Society*, pp. 150-159, Taylor & Francis, London.

[16] Kolski C. & Millot P. (1991). A rule-based approach for the ergonomic evaluation of man-machine graphic interface. *International Journal of Man-Machine Studies*, 35, pp. 657-674.

[17] Moussa F. & Kolski C. (1992). Vers une formalisation d'une démarche de conception de synoptiques industriels: application au système ERGO-CONCEPTOR. *Proceedings Colloque ERGO-IA Ergonomie et Informatique Avancée*, 7-9, October, Biarritz, France.

[18] Myers B.A. (1995). User interface software tools. *ACM Transactions on Computer-Human Interaction*, 2 (1), pp. 64-103, March.

[19] Szekely P. (1996) Retrospective and Challenges for Model-Based Interface Development. In J. Vanderdonckt (ed.), *Proceedings of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96* (Namur, 5-7 June 1996), Presses Universitaires de Namur, Namur, pp. xxi-xliv.

[20] Van Daele A. (1992). *La réduction de la complexité par les opérateurs dans le contrôle de processus continus, contribution à l'étude du contrôle par anticipation et de ses conditions de mise en oeuvre*. Ph.D. Dissertation in Psychology, university of Liège, Belgium, october.

[21] Rouse W.B. (1983). Models of human problem solving : detection, diagnosis and compensation for system failures. *Automatica*, 19 (6), pp. 613-625, november.

[22] Taborin V. & Millot P. (1989). Cooperation between man and decision aid system in supervisory loop of continuous processes. *Proceedings 8th European Annual Conference on Human Decision Making and Manual Control*, Lyngby, Denmark, June 12-14.

[23] Boy G. (1991). *Intelligent assistant systems*. New York: Academic Press.

[24] Norman D.A. (1986). Cognitive engineering. In D.A. Norman & S.W Draper (Eds), *User centred system design : new perspectives on human computer interaction*, pp. 31-61, Hillsdale, NJ : Erlbaum.

[25] Leplat J. (1985). *Erreur humaine, fiabilité humaine dans le travail*. Armand Colin, Paris.

[26] Reason J. (1990). *Human error*. Cambridge University Press.

[27] Senders J.W. & Moray N. (1991). *Human error, cause, prediction and reduction*. Lawrence Erlbaum Associates Publishers.

[28] Kolski C., Tendjoui M. & Millot P. (1992). A process method for the design of intelligent man-machine interfaces: case study: the Decisional Module of Imagery. *International Journal of Human Factors in Manufacturing*, vol. 2 (2), pp. 155-175.

[29] Millot P. & Debernard S. (1993). Men-machines cooperative organizations: methodologicial and practical attemps in air traffic control. *Proceedings IEEE Conference on Systems, Man and Cybernetics*, Le Touquet, France, volume 1, pp. 695-700, October 17-20.

[30] Abed M. & Angue J.C. (1994). A new method for conception, realisation and evaluation of man-machine. *Proceedings IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, Texas, October 2-5.

[31] Daniellou F. (1986). *L'opérateur, la vanne, l'écran: l'ergonomie dans les salles de contrôle*. Montrouge, ANACT, collection "Outils et Méthodes".

[32] Ivergard T. (1989). *Handbook of control room design and ergonomics*. Taylor and Francis, London.

[33] Stanton N.A. (1995). *Human Factors in nuclear safety*. Taylor & Francis, London.

[34] Villemeur A. (1988). *Sûreté de fonctionnement des systèmes industriels : fiabilité, facteur humain, informatisation*. Paris : Eyrolles.

[35] Ross D.T., Kenneth E. & Schoman J.R. (1977). Structured analysis for requirements definition. *IEEE Transactions on Software engineering*, vol. SE-3, 1, January.

[36] IGL Technology (1989). *SADT, un langage pour communiquer*. Eyrolles, Paris.

[37] DeMarco T. (1979). *Structured analysis and system specification*. Yourdon Press, NY, Englewood Cliffs, Prentice Hall.

[38] Sinclair I.A.C., Sell R.G., Beishon R.J. & Bainbridge L. (1965). Ergonomic study of L.D. Waste-heat boiler control room. *Journal Iron and steel inst.*, 204, pp. 434-442.

[39] Recht J.L. (1966). *Failure mode and effect*. National Safety Council, USA.

[40] Stammers R. B., Carey M. S. & Astley J. A. (1990). Task analysis. In *Evaluation of human work. A Practical Ergonomics Methodology*, Wilson J. R. & Corlett E. N. (Eds.), Taylor & Francis.

[41] Kirwan B. (1994). *A guide to practical human reliability assessment*. Taylor & Francis.

[42] Nielsen J. (1993). *Usability engineering*. Academic Press.

[43] Wilson J.R. & Corlett E.N. (1995). *Evaluation of human work, a practical ergonomics methodology, 2nd edition*. Taylor and Francis, London.

[44] Jordan P.W. & Thomas B. (1996). *Usability evaluation in industry*. Taylor & Francis, London.

[45] Jacob R.J.K. (1986). A specification language for direct-manipulation user interfaces. *ACM Transactions on Graphics*, 5 (4), pp. 283-317, October.

[46] Olsen D.R. (1983). MIKE: the Menu Interaction Kontrol Environment. *ACM Transactions on Information systems*, 5 (4), pp. 318-344.

[47] Singh G. & Green M. (1991). Automating the lexical and syntactic design of graphical user interfaces: the Uofa* UIMS. *ACM Transactions on Graphics*, 10 (3), pp. 213-254, July.

[48] Bullinger H.J., Fahnrich K.P. & Weisbecker A. (1996). GENIUS: Generating software-ergonomic user interfaces. *International Journal of Human-Computer Interaction*, 8 (2), pp. 115-144.

[49] Vanderdonckt J. (ed.) (1996). *Proceedings of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96* (Namur, 5-7 June 1996), Presses Universitaires de Namur, Namur.

[50] Kim W.C. & Foley J. (1993). Providing high-level control and expert assistance in the user interface presentation design. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel & T. White (Eds.), *Proceedings of the Conference on Human Factors in Computing Systems, INTERCHI'93, "Bridges between worlds"*, pp. 430-437, New-York: ACM.

[51] Szekely P., Luo P. & Neches R. (1992). Faciliting the exploration of interface design alternatives: the HUMANOID model of interface design. In *CHI'92 ACM Conference on Human factors in Computing Systems*, Bauersfeld P., Bennett J. & Lynch G. (Eds.), Monterey, California, pp. 507-515, May 3-7, New York: ACM Press.

[52] Schlungbaum E. & Elwer T. (1996). Automatic user interface generation from declarative models. In Vanderdonckt J. (1996), *Proceedings of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96* (Namur, 5-7 June 1996), Presses Universitaires de Namur, Namur.

[53] Puerta A.R. (1996). The Mecano project: comprehensive and integrated support for model-based interface development. In Vanderdonckt J. (Ed.), *Proceedings of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96* (Namur, 5-7 June 1996), Presses Universitaires de Namur, Namur.

[54] Puerta A.R. (1997). A Model-Based Interface Development Environment. *IEEE Software*, Vol. 41, No. 4, July/August, pp. 40-47.

[55] Vanderdonckt J. & Bodart F. (1993). Encapsulating knowledge for intelligent automatic interaction objects selection. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel & T. White (Eds.), *Proceedings of the Conference on Human Factors in Computing Systems, INTERCHI'93, "Bridges between worlds"*, pp. 424-429, New-York: ACM.

[56] Sukaviriya P., Foley J. & Griffith T. (1993). A second generation user interface design environment: the model and the runtime architecture. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel & T. White (Eds.), *Proceedings of the Conference on Human Factors in Computing Systems, INTERCHI'93, "Bridges between worlds"*, pp. 375-382, New-York: ACM

[57] Balzert H. (1995). From OOA to GUI - The JANUS system. In S.A. Arnesen, D. Gilmore, K. Nordby & P. Helmersen (Eds.), *Proceedings of the Fifth IFIP TC 13 Conference on Human-Computer Interaction INTERACT'95*, pp. 319-324, London: Chapman & Hall.

[58] Moussa F. & Moalla M. (1995). Toward a user centered system design methodology: Application to the graphical interfaces design". *AMSE Periodicals "Advanced in modelling and analysis"*, Vol. 35, N°1, pp. 1-10.

[59] Coekin J.A. (1968). A versatile presentation of parameters for rapid recognition of total state. *Proceedings International Symposium on Man-Machine Systems*, IEEE Conference Record 69 C58-MMS, September.

[60] Schmid C.F. & Schmid S.E. (1979). *Handbook of graphic presentation, 2nd Edition*. McGraw-Hill, New York.

[61] ILOG (1994). *MASAÏ and AIDA reference Manual, version 1.76*. ILOG S.A., 2, avenue Galliéni, BP85, 94253 Gentilly cedex, France.

[62] Rasmussen J. (1985). The role of hierarchical knowledge representation in decision-making and system management. *IEEE Transactions on Systems, Man and Cybernetics*, 15 (2), pp. 234-243.

[63] Lind M. (1990). *Representing goals and fonctions of complex systems: an introduction to Multilevel Flow Modelling*. RISO Laboratory, Denmark, Ref: 90-D-381. ISBN 87-87950-52-9.

[64] Rasmussen J., Petersen A.M. & Goodstein L.P. (1994). *Cognitive systems engineering*. Wiley Interscience, New-York.

[65] Bisantz A.M. & Vicente K. (1994). Making the abstraction hierarchy concrete. *International Journal of Human-Computer Interaction*, 40, pp. 83-117.

[66] Rasmussen J. (1980). The human as a system component. In H.T. Smith and T.R.G. Green (Eds.), *Human Interaction with Computer*, pp. 67-96, London Academic Press.

[67] Lind M. (1993), Modeling goals and functions of complex industrial plant. *Proceedings AAAI-93 Workshop on reasoning about Functions*, Washington D.C., July, 11.

[68] Goodstein L.P. (1982). An integrated display set for process operators. *Proceedings IFAC Congress Analysis, Design and Evaluation of Man-Machine Systems*, Baden-Baden, September.

[69] Javaux D., Colard M.I. & Vanderdonckt J. (1996). Visual display design: a comparison of two methodologies. *Proceedings 1st International Conference on Applied Ergonomics ICAE'96*, USA Publishing, West Lafayette, 1996, pp. 662-667.

[70] Moussa F. (1992). *Contribution à la conception ergonomique des interfaces de supervision dans les procédés industriels: Application au système ERGO-CONCEPTOR*. Unpublished Ph.D. Dissertation, University of Valenciennes, France.

[71] Smith S.L., Mosier J.N. (1986). *Guidelines for designing user interface software*. Report EDS-TR-86-278, The MITRE Corporation, Bedford, MA.

[72] Shneiderman B. (1987). *Designing the user Interface, Strategies for effective human-computer interaction*. Addison Wesley Publishing Compagny.

[73] Vanderdonckt J. (1994a). *Guide ergonomique des interfaces homme-machine*. Presses Universitaires de Namur, Namur.

[74] Vanderdonckt J. (1994b). Jusqu'au bout avec nos règles ergonomiques. *Proceedings IHM'94, Sixièmes Journées sur l'Ingénierie des Interfaces Homme-Machine*, Lille, France, 8-9 december, pp. 231-236.

[75] Bastien J.M.C. & Scapin D.L. (1992). A validation of ergonomic criteria for the evaluation of human-computer interfaces. *International Journal of Human-Computer Interaction*, 4 (2), pp. 183-196.

[76] Scapin D. (1990). Organizing human factors knowledge for the evaluation and design of interfaces. *International Journal of Human-Computer Interaction*, 2 (3), pp. 203-229.

[77] Bastien J.M.C. & Scapin D.L. (1995). Evaluating a user interface with ergonomic criteria. *International Journal of Human-Computer Interaction*, 7 (2), pp. 105-121.

[78] Vanderdonckt J. (1995). *Tools for working with guidelines, tutorial Notes*. Tutorial 12, HCI'95, Sixth International Conference on Human-Computer Interaction, Pacifico Yokohama, Pacific Convention Plaza, Yokohama, Japan, 10 July.

[79] Mackinlay J. (1986). Automating the design of graphical presentation of relationnal information. *ACM Transactions on Graphics*, 5 (2), pp. 110-141.

[80] Perlman G. (1987). An axiomatic model of information presentation. *Proceedings of the 1987 Human Factors Society Meeting*, New York, Human Factors Society.

[81] Martin C. (1990). A UIMS for knowledge based interface template generation and interaction. In D. Diaper, D. Gilmore, G. Cockton & B. Shackel (Eds.), *Proceedings of the IFIP TC 13 Conference on Human-Computer Interaction, Interact'90*, pp. 651-657, Amsterdam: North Holland.

[82] Wiecha C., Bennett W., Boises S., Gould j. & Green S. (1990). ITS: A tool for rapidly developing interactive applications. *ACM Transactions on Information Systems*, 8, pp. 204-236.

[83] Bebin J. (1984). Génie Nucléaire. *Techniques de l'Ingénieur*, Edition N°1063, May, France.

[84] Woods D.D., Wise J.A. & Hanes L.F. (1981). An evaluation of nuclear power plant safety parameter display systems. *25th Annual Meeting Human Factors Society*, pp. 110-114.

[85] Kolski C. & Moussa F. (1996). Two examples of "Tools for working with guidelines" for process control field: SYNOP and ERGO-CONCEPTOR. *Third Annual Meeting of the International Special Interest Group on "Tools for Working with Guidelines"*, J. Vanderdonckt (Ed.), Namur, Belgium, June 4.

[86] Tullis T.S. (1988). Screen design. In *Handbook of Human-Computer Interaction*, M. Helander (Ed.), pp. 377-411, Elsevier Science Publishers B.V., North-Holland.

[87] Riahi M., Moussa F., Moalla M. & Kolski C. (1998). Approche de spécification formelle des interfaces homme-machine par réseaux de Petri interprétés dans les procédés industriels. *Proceedings ERGO'IA International Conference*, November 3-6, Biarritz, France.

[88] Cohen A., Crow D., Dili I., Gorny P., Hoffman H.J., Iannella R., Ogawa K., Reiterer H., Ueno K. & Vanderdonckt J. (1995). Tools for Working with guidelines. *SIGCHI*, 27 (2), pp. 30-32, ACM Press.